

Entwicklerhandbuch für Version 2.x

# AWS SDK for Java 2.x



# AWS SDK for Java 2.x: Entwicklerhandbuch für Version 2.x

Copyright © 2024 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

Entwicklerhandbuch — AWS SDK for Java 2.x .....	1
Fangen Sie mit dem SDK an .....	1
Entwickeln Sie mobile Anwendungen .....	1
Wartung und Support für SDK-Hauptversionen .....	1
Weitere Ressourcen .....	2
Tragen Sie zum SDK bei .....	2
Erste-Schritte-Tutorial .....	3
Schritt 1: Richten Sie sich für dieses Tutorial ein .....	3
Schritt 2: Erstellen Sie das Projekt .....	3
Schritt 3: Schreiben Sie den Code .....	8
Schritt 4: Erstellen Sie die Anwendung und führen Sie sie aus .....	12
Herzlichen Glückwunsch .....	13
Bereinigen .....	13
Nächste Schritte .....	14
Aufstellen .....	15
Übersicht über die Einrichtung .....	15
Anmeldefähigkeit für das - AWS Zugriffsportal .....	16
Einrichten des Single-Sign-On-Zugriffs für das SDK .....	16
Anmelden mit der AWS CLI .....	17
Installieren von Java und eines Build-Tools .....	18
Zusätzliche Authentifizierungsoptionen .....	18
Richten Sie ein Apache Maven-Projekt ein .....	19
Voraussetzungen .....	19
Erstellen eines Maven-Projekts .....	19
Konfigurieren des Java-Compilers für Maven .....	20
Deklarieren des SDK als Abhängigkeit .....	21
Festlegen von Abhängigkeiten für SDK-Module .....	22
Erstellen Ihres Projekts .....	24
Richten Sie ein Gradle-Projekt ein .....	25
Richten Sie ein GraalVM Native Image-Projekt ein .....	32
Voraussetzungen .....	32
Create a project using the archetype .....	32
Erstellen Sie ein natives Image .....	33
Verwenden des SDK .....	34

Arbeiten mit Service-Clients .....	34
Erstellen eines Service-Clients .....	34
Standard-Client-Konfiguration .....	35
Konfigurieren von Service-Clients .....	35
Anfragen stellen .....	36
Antworten verarbeiten .....	36
Schließen des Service-Clients .....	37
Ausnahmen behandeln .....	38
Verwenden von Waitern .....	39
HTTP-Clients .....	39
Wiederholversuche .....	39
Timeouts .....	39
Ausführungsabfänger .....	40
Zusätzliche Informationen .....	40
Bereitstellen temporärer Anmeldeinformationen für das SDK .....	40
Konfigurieren des Zugriffs auf temporäre Anmeldeinformationen .....	41
Standard-Anbieterkette für Anmeldeinformationen .....	43
Verwenden eines bestimmten Anbieters von Anmeldeinformationen oder einer bestimmten Anbieterkette .....	44
Verwenden von Profilen .....	45
Laden temporärer Anmeldeinformationen aus einem externen Prozess .....	48
Temporäre Anmeldeinformationen im Code bereitstellen .....	52
Lesen von IAM-Rollenanmeldeinformationen auf Amazon EC2 .....	55
Verwenden von AWS-Regionen .....	57
Explizite Konfiguration eines AWS-Region .....	57
Bestimmen der Region aus der Umgebung .....	58
Überprüfen der Serviceverfügbarkeit .....	59
Auswählen eines bestimmten Endpunkts .....	60
Reduzieren Sie die Startzeit des SDK für AWS Lambda .....	60
Verwenden der SDKs <code>URLConnectionHttpClient</code> .....	60
Verwenden der SDKs <code>AwsCrtAsyncHttpClient</code> .....	61
Entfernen ungenutzter HTTP-Client-Abhängigkeiten .....	61
Konfigurieren von Service-Clients für die Verknüpfung von Suchvorgängen .....	63
Initialisieren des SDK-Clients außerhalb des Lambda-Funktionshandlers .....	64
Minimieren der Abhängigkeitsinjektion .....	64
Verwenden eines Maven-Arche-Typ-Targetings AWS Lambda .....	65



Lambda SnapStart .....	65
Änderungen der Version 2.x, die sich auf die Startzeit auswirken .....	65
Weitere Ressourcen .....	65
HTTP-Clients .....	66
Verfügbare Clients .....	66
Empfehlungen von Kunden .....	67
Intelligente Standardeinstellungen .....	70
Proxy-Unterstützung .....	72
Den Apache-basierten HTTP-Client konfigurieren .....	75
Den URL-verbindungs-basierten HTTP-Client konfigurieren .....	79
Konfigurieren des Netty-basierten HTTP-Clients .....	86
AWSCRT-basierte HTTP-Clients konfigurieren .....	91
Ausnahmeverarbeitung .....	102
Warum nicht aktivierte Ausnahmen? .....	102
AwsServiceException (und Unterklassen) .....	103
SdkClientException .....	104
Ausnahmen und Wiederholungsverhalten .....	104
Protokollierung .....	104
Log4j 2-Konfigurationsdatei .....	104
Protokollierungsabhängigkeit hinzufügen .....	105
SDK-spezifische Fehler und Warnungen .....	106
Protokollierung der Anforderungs-/Antwortübersicht .....	106
Verbose-Wire-Protokollierung .....	107
Festlegen der JVM-TTL für DNS-Namenssuche .....	113
Festlegen der JVM-TTL .....	113
Bewährte Methoden .....	114
Einen SDK-Client wiederverwenden .....	114
Schließt Eingabestreams .....	114
Optimieren Sie HTTP-Konfigurationen .....	115
Verwenden Sie OpenSSL für Netty .....	115
API-Timeouts konfigurieren .....	116
Verwenden Sie Metriken .....	117
SDK-Funktionen verwenden .....	118
Allgemeine Funktionen .....	118
Servicespezifische Funktionen .....	118
Arbeiten mit paginierten Ergebnissen .....	119

Synchrone Paginierung .....	119
Asynchrone Paginierung .....	122
Abfrage nach Ressourcenstatus .....	128
Voraussetzungen .....	128
Verwenden von Waitern .....	128
Konfigurieren von Waitern .....	129
Codebeispiele .....	130
Asynchrone Programmierung verwenden .....	131
Nicht-Streaming-Operationen .....	131
Streaming-Operationen .....	134
Erweiterte Operationen .....	138
Arbeiten mit HTTP/2 .....	139
SDK-Metriken verwenden .....	139
Voraussetzungen .....	140
So aktivieren Sie die Erfassung von Metriken .....	141
Wann sind Metriken verfügbar? .....	142
Welche Informationen werden erfasst? .....	142
Wie kann ich diese Informationen verwenden? .....	142
Service-Client-Metriken .....	143
Arbeiten mit AWS-Services .....	149
CloudWatch .....	149
Abrufen von Metriken von CloudWatch .....	150
Veröffentlichen von benutzerdefinierten Metrikdaten in CloudWatch .....	152
Arbeiten mit CloudWatch Alarmen .....	154
Verwenden von Amazon CloudWatch Events .....	158
AWS -Datenbankservices .....	162
Amazon DynamoDB .....	162
Amazon RDS .....	163
Amazon Redshift .....	163
Amazon Aurora Serverless v1 .....	164
Amazon DocumentDB .....	164
DynamoDB .....	164
Arbeiten mit Tabellen in DynamoDB .....	165
Arbeiten mit Elementen in DynamoDB .....	175
Objekte DynamoDB-Elementen zuordnen .....	182
Amazon EC2 .....	304

Amazon EC2 Instanzen verwalten .....	305
Verwenden von AWS-Regionen und Availability Zones .....	311
Arbeiten mit Sicherheitsgruppen in Amazon EC2 .....	315
Mit Amazon EC2 EC2-Instance-Metadaten arbeiten .....	320
IAM .....	326
Verwalten von IAM Zugriffsschlüsseln .....	327
IAM Benutzer verwalten .....	333
Erstellen von IAM-Richtlinien .....	337
Arbeiten mit IAM Richtlinien .....	345
Arbeiten mit IAM Serverzertifikaten .....	352
Kinesis .....	356
Abonnieren von Amazon Kinesis Data Streams .....	357
Lambda .....	368
Aufrufen von einer Lambda-Funktion .....	368
Auflisten von Lambda-Funktionen .....	369
Löschen einer Lambda-Funktion .....	370
Amazon S3 .....	371
Verwenden von Zugriffspunkten oder Multi-Regions-Zugriffspunkten .....	372
Bucket-Operationen .....	373
Objektoperationen .....	380
Vorsignierte URLs .....	390
Regionsübergreifender Zugriff .....	399
Prüfsummen .....	400
Verwenden Sie einen performanten S3-Client .....	402
Übertragen von Dateien und Verzeichnissen .....	405
Amazon SNS .....	413
Erstellen eines Themas .....	413
Listen Sie Ihre Amazon SNS Themen auf .....	414
Abonnieren eines Endpunkts für ein Thema .....	415
Veröffentlichen einer Nachricht für ein Thema .....	416
Abmelden eines Endpunkts von einem Thema .....	417
Löschen eines Themas .....	418
Amazon SQS .....	419
Warteschlangenoperationen .....	419
Nachrichtenoperationen .....	423
Amazon Transcribe .....	427

Richten Sie das Mikrofon ein .....	427
Erstellen Sie einen Herausgeber .....	428
Erstellen Sie den Client und starten Sie den Stream .....	430
Weitere Informationen .....	426
Codebeispiele .....	433
Aktionen und Szenarien .....	433
API Gateway .....	435
Application Auto Scaling .....	439
Application Recovery Controller .....	448
Aurora .....	451
Auto Scaling .....	486
Amazon Bedrock .....	549
Amazon-Bedrock-Laufzeit .....	554
CloudFront .....	580
CloudWatch .....	600
CloudWatch Ereignisse .....	651
CloudWatch Protokolle .....	657
Amazon Cognito Identity .....	668
Amazon Cognito Identity Provider .....	676
Amazon Comprehend .....	703
DynamoDB .....	714
Amazon EC2 .....	781
Amazon ECS .....	853
Elastic Load Balancing .....	867
MediaStore .....	912
OpenSearch Service .....	927
EventBridge .....	936
Forecast .....	968
AWS Glue .....	981
HealthImaging .....	1005
IAM .....	1031
AWS IoT .....	1117
AWS IoT data .....	1145
Amazon Keyspaces .....	1147
Kinesis .....	1174
AWS KMS .....	1187

Lambda .....	1206
MediaConvert .....	1230
Migration Hub .....	1253
Amazon Personalize .....	1266
Amazon Personalize Events .....	1297
Amazon Personalize Runtime .....	1300
Amazon Pinpoint .....	1305
Amazon-Pinpoint-SMS- und -Sprachnachrichten-API .....	1350
Amazon Polly .....	1354
Amazon RDS .....	1360
Amazon Redshift .....	1400
Amazon Rekognition .....	1406
Route-53-Domainregistrierung .....	1475
Amazon S3 .....	1497
S3 Glacier .....	1605
SageMaker .....	1621
Secrets Manager .....	1650
Amazon SES .....	1664
Amazon SES-API v2 .....	1676
Amazon SNS .....	1680
Amazon SQS .....	1729
Step Functions .....	1750
AWS STS .....	1774
AWS Support .....	1777
Systems Manager .....	1801
Amazon Textract .....	1810
Amazon Transcribe .....	1821
Serviceübergreifende Beispiele .....	1837
Erstellen Sie eine App zum Senden von Daten an eine DynamoDB-Tabelle .....	1838
Erstellen eines Amazon Lex-Chatbots .....	1838
Erstellen einer Amazon-SNS-Anwendung .....	1839
Erstellen einer Messaging-Anwendung .....	1839
Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos .....	1840
Erstellen einer Webanwendung zur Verfolgung von DynamoDB-Daten .....	1840
Erstellen einer Webanwendung zur Verfolgung von Amazon-Redshift-Daten .....	1841
Erstellen Sie einen Tracker für Aurora-Serverless-Arbeitsaufgaben .....	1841

Erstellen einer Anwendung zum Analysieren von Kundenfeedback .....	1842
Erkennen von PSA in Bildern .....	1842
Erkennen von Objekten in Bildern .....	1843
Erkennen Sie Personen und Objekte in einem Video .....	1843
Veröffentlichen Sie Nachrichten in Warteschlangen .....	1844
Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion .....	1844
Verwenden von Step Functions, um Lambda-Funktionen aufzurufen .....	1845
Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion .....	1845
Sicherheit .....	1847
Datenschutz .....	1847
Transport Layer Security (TLS) .....	1849
TLS-Versionen überprüfen .....	1849
TLS-Versionen erzwingen .....	1850
Migrieren zu TLS 1.2 .....	1850
Identitäts- und Zugriffsverwaltung .....	1850
Zielgruppe .....	1851
Authentifizierung mit Identitäten .....	1851
Verwalten des Zugriffs mit Richtlinien .....	1855
Funktionsweise AWS-Services von mit IAM .....	1858
Fehlerbehebung für AWS Identität und Zugriff .....	1858
Compliance-Validierung .....	1860
Ausfallsicherheit .....	1862
Sicherheit der Infrastruktur .....	1862
Migrieren zu Version 2 .....	1864
Was ist neu in Version 2? .....	1864
S-tep-by-step Anweisungen .....	1865
Übersicht über die Schritte .....	1865
Beispielmigration .....	1867
Was unterscheidet sich zwischen 1.x und 2.x? .....	1878
Änderung des Paketnamens .....	1878
Hinzufügen von Version 2.x zu Ihrem Projekt .....	1879
Unveränderliche POJOs .....	1879
Setter- und Getter-Methoden .....	1880
Modellklassennamen .....	1880
Bibliotheken und Dienstprogramme .....	1881
Client-Änderungen .....	1883

---

Änderungen des Anmeldeinformationsanbieters .....	1927
Änderungen der Region .....	1935
Änderungen an Vorgängen, Anforderungen und Antworten .....	1937
Ausnahmeänderungen .....	1939
Serialisierungsänderungen .....	1940
Servicespezifische Änderungen .....	1941
Änderungen der Profildatei .....	1947
Externe Konfiguration .....	1948
Waiter .....	1951
S3 Transfer Manager .....	1955
EC2-Metadaten-Hilfsprogramm .....	1962
CloudFront Vorsignieren .....	1970
S3-URI-Parsing .....	1973
Verwenden Sie das SDK for Java 1.x und 2.x side-by-side .....	1976
OpenPGP-Schlüssel .....	1978
Aktueller Schlüssel .....	1978
Dokumentverlauf .....	1980
.....	mcmIxxxviii

# Entwicklerhandbuch — AWS SDK for Java 2.x

Das AWS SDK for Java bietet eine Java-API für AWS-Services. Mit dem SDK können Sie Java-Anwendungen erstellen, die mit Amazon S3, Amazon EC2, Amazon DynamoDB, und mehr funktionieren.

AWS SDK for Java 2.x ist eine grundlegende Neufassung der Codebasis von Version 1.x. Sie basiert auf Java 8+ und fügt mehrere häufig angeforderte Funktionen hinzu. Dazu gehören die Unterstützung für blockierungsfreie I/O und die Möglichkeit, zur Laufzeit eine andere HTTP-Implementierung einzubinden.

Unterstützung für neue Services ergänzen wir regelmäßig im AWS SDK for Java. Eine Liste der Änderungen und Funktionen in einer bestimmten Version finden Sie im [Änderungsprotokoll](#).

## Fangen Sie mit dem SDK an

Wenn Sie bereit sind, das SDK in der Praxis auszuprobieren, folgen Sie dem [Erste-Schritte-Tutorial](#) Tutorial.

Informationen zum Einrichten Ihrer Entwicklungsumgebung finden Sie unter [Aufstellen](#).

Wenn Sie derzeit Version 1.x von verwenden SDK for Java, finden Sie spezifische Anleitungen unter [Migration zu Version 2](#).

Informationen zum Stellen von Anfragen an Amazon EC2 und zu Amazon S3 anderen AWS-Services finden Sie unter [Verwenden](#) von SDK for Java und [Arbeiten mit AWS-Services](#). DynamoDB

## Entwickeln Sie mobile Anwendungen

Wenn Sie ein Entwickler mobiler Apps sind, Amazon Web Services stellt das [AWS Amplify](#) Framework bereit.

## Wartung und Support für SDK-Hauptversionen

Informationen zu Wartung und Support für SDK-Hauptversionen und deren zugrunde liegende Abhängigkeiten finden Sie in den folgenden Themen im [Referenzhandbuch zu AWS SDKs und Tools](#):

- [AWS-Wartungsrichtlinie für SDKs und Tools](#)



- [AWSMatrix zur Versionsunterstützung von SDKs und Tools](#)

## Weitere Ressourcen

Zusätzlich zu diesem Handbuch stehen AWS SDK for Java-Entwicklern folgende wertvolle Online-Ressourcen zur Verfügung:

- [AWS SDK for Java2.x API-Referenz](#)
- [Java-Entwicklerblog](#)
- [Thema zur Java-Entwicklung in AWS re:Post](#)
- [SDK-Quelle](#) aktiviert GitHub
- [AWSBibliothek mit SDK-Codebeispielen](#)
- [@awsforjava \(Twitter\)](#)

## Tragen Sie zum SDK bei

Entwickler können ihr Feedback auch über die folgenden Kanäle senden:

- Probleme einreichen am GitHub:
  - [Probleme mit der Dokumentation im Entwicklerhandbuch einreichen](#)
  - [Übermitteln von SDK-Problemen](#)
- Nehmen Sie auf dem AWS SDK for Java [2.x-Gitter-Channel](#) an einem informellen Chat über das SDK teil

# Erste Schritte mit dem AWS SDK for Java 2.x

Die AWS SDK for Java 2.x stellt Java-APIs für Amazon Web Services (AWS) bereit. Mit dem SDK können Sie Java-Anwendungen erstellen, die mit Amazon S3, Amazon EC2, Amazon DynamoDB, und mehr funktionieren.

Dieses Tutorial zeigt Ihnen, wie Sie [Apache Maven](#) verwenden, um Abhängigkeiten für das SDK for Java 2.x zu definieren und dann Code zu schreiben, der eine Verbindung herstellt, Amazon S3 um eine Datei hochzuladen.

Folgen Sie diesen Schritten, 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: Richten Sie sich für dieses Tutorial ein

Bevor Sie mit diesem Tutorial beginnen, benötigen Sie Folgendes:

- Erlaubnis zum Zugriff Amazon S3
- Eine Java-Entwicklungsumgebung, die so konfiguriert ist, dass sie AWS-Services per Single Sign-On auf die AWS IAM Identity Center

Verwenden Sie die Anweisungen unter [???](#), um sich auf dieses Tutorial vorzubereiten. Nachdem Sie [Ihre Entwicklungsumgebung mit Single Sign-On-Zugriff für das Java SDK konfiguriert](#) haben und eine [aktive AWS Access-Portal-Sitzung](#) eingerichtet haben, fahren Sie mit Schritt 2 dieses Tutorials fort.

## Schritt 2: Erstellen Sie das Projekt

Um das Projekt für dieses Tutorial zu erstellen, führen Sie einen Maven-Befehl aus, der Sie zur Eingabe der Konfiguration des Projekts auffordert. Nachdem alle Eingaben eingegeben und bestätigt wurden, beendet Maven den Aufbau des Projekts durch die Erstellung von Java-Stub-Dateien `pom.xml` und erstellt diese.

1. Öffnen Sie ein Terminal oder ein Befehlszeilenfenster und navigieren Sie zu einem Verzeichnis Ihrer Wahl, z. B. Ihrem Ordner Desktop oder Home.
2. Geben Sie im Terminal den folgenden Befehl ein und drücken Sie `Enter`.

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

3. Geben Sie für jede Eingabeaufforderung den in der zweiten Spalte aufgeführten Wert ein.

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

4. Nachdem der letzte Wert eingegeben wurde, listet Maven die von Ihnen getroffenen Entscheidungen auf. Bestätigen Sie, indem Sie Werte eingeben, oder geben Sie sie erneut ein, indem Sie sie eingeben. *N*

Maven erstellt den Projektordner, der auf dem von Ihnen `artifactId` eingegebenen Wert `getstarted` basiert. Suchen Sie im `getstarted` Ordner nach einer `README.md` Datei, die Sie überprüfen können, nach einer `pom.xml` Datei und einem `src` Verzeichnis.

Maven erstellt den folgenden Verzeichnisbaum.

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

10 directories, 7 files
```

Im Folgenden wird der Inhalt der `pom.xml` Projektdatei gezeigt.

## **pom.xml**

Der `dependencyManagement` Abschnitt enthält eine Abhängigkeit von AWS SDK for Java 2.x und der `dependencies` Abschnitt hat eine Abhängigkeit von Amazon S3. Das Projekt verwendet Java 1.8 aufgrund des 1.8 Werts in den `maven.compiler.target` Eigenschaften `maven.compiler.source` und.

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

```

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

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

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

```

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

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

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

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

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

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

```
</dependency>

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

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

</project>
```

## Schritt 3: Schreiben Sie den Code

Der folgende Code zeigt die von Maven erstellte App Klasse. Die `main` Methode ist der Einstiegspunkt in die Anwendung, die eine Instanz der `Handler` Klasse erstellt und dann ihre `sendRequest` Methode aufruft.

### App-Klasse

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

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

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

        Handler handler = new Handler();
```

```
        handler.sendRequest();

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

Die von Maven erstellte `DependencyFactory` Klasse enthält die `s3Client` Factory-Methode, die eine [S3Client](#) Instanz erstellt und zurückgibt. Die `S3Client` Instanz verwendet eine Instanz des Apache-basierten HTTP-Clients. Dies liegt daran, dass Sie angegeben haben `apache-client`, als Maven Sie nach dem zu verwendenden HTTP-Client gefragt hat.

Das `DependencyFactory` wird im folgenden Code gezeigt.

## DependencyFactory-Klasse

```
package org.example;

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

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

    private DependencyFactory() {}

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

Die `Handler` Klasse enthält die Hauptlogik Ihres Programms. Wenn in der `App` Klasse eine Instanz von `Handler` erstellt wird, stellt `DependencyFactory` sie den `S3Client` Service-Client bereit. Ihr Code verwendet die `S3Client` Instance, um den Amazon S3 `S3-Service` aufzurufen.



Maven generiert die folgende `Handler` Klasse mit einem `TODO` Kommentar. Der nächste Schritt im Tutorial ersetzt den `TODO` mit Code.

## Handler Klasse, von Maven generiert

```
package org.example;

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

public class Handler {
    private final S3Client s3Client;

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

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

Um die Logik auszufüllen, ersetzen Sie den gesamten Inhalt der `Handler` Klasse durch den folgenden Code. Die `sendRequest` Methode wird ausgefüllt und die erforderlichen Importe werden hinzugefügt.

## Handler Klasse, implementiert

Der Code erstellt zunächst einen neuen S3-Bucket, wobei der letzte Teil des Namens generiert wird, um den Bucket-Namen eindeutig zu machen. `System.currentTimeMillis()`

Nach dem Erstellen des Buckets in der `createBucket()` Methode lädt das Programm ein Objekt mit der [putObject](#) Methode von `S3Client` hoch. Der Inhalt des Objekts ist eine einfache Zeichenfolge, die mit der `RequestBody.fromString` Methode erstellt wurde.

Schließlich löscht das Programm das Objekt, gefolgt vom Bucket in der `cleanup` Methode.

```
package org.example;

import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
```

```
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

public class Handler {
    private final S3Client s3Client;

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

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

        createBucket(s3Client, bucket);

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

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

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

        cleanUp(s3Client, bucket, key);

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

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

```
s3Client.waiter().waitUntilBucketExists(HeadBucketRequest.builder()
    .bucket(bucketName)
    .build());
System.out.println(bucketName + " is ready.");
System.out.printf("%n");
} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

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

## Schritt 4: Erstellen Sie die Anwendung und führen Sie sie aus

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

1. Stellen Sie sicher, dass Sie über eine aktive IAM Identity Center-Sitzung verfügen. Führen Sie dazu den AWS Command Line Interface Befehl aus `aws sts get-caller-identity` und

überprüfen Sie die Antwort. Wenn Sie keine aktive Sitzung haben, finden Sie in [diesem Abschnitt](#) Anweisungen.

- Öffnen Sie ein Terminal- oder Befehlszeilenfenster und navigieren Sie zu Ihrem Projektverzeichnisgetstarted.
- Verwenden Sie den folgenden Befehl, um Ihr Projekt zu erstellen:

```
mvn clean package
```

- Verwenden Sie den folgenden Befehl, um die Anwendung auszuführen.

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

Gehen Sie wie folgt vor, um den neuen Bucket und das neue Objekt anzuzeigen, die das Programm erstellt.

- Kommentieren Sie die Zeile `cleanup(s3Client, bucket, key)` in der `sendRequest` Methode aus und speichern Sie die Datei. `Handler.java`
- Erstellen Sie das Projekt neu, indem Sie es ausführen`mvn clean package`.
- Führen Sie `mvn exec:java -Dexec.mainClass="org.example.App"` den Vorgang erneut aus, um das Textobjekt erneut hochzuladen.
- Melden Sie sich bei [der S3-Konsole](#) an, um das neue Objekt im neu erstellten Bucket anzuzeigen.

Nachdem Sie sich die Datei angesehen haben, löschen Sie das Objekt und anschließend den Bucket.

## Herzlichen Glückwunsch

Wenn Ihr Maven-Projekt ohne Fehler erstellt und ausgeführt wurde, dann herzlichen Glückwunsch! Sie haben erfolgreich Ihre erste Java-Anwendung mit dem SDK for Java 2.x erstellt.

## Bereinigen

Gehen Sie wie folgt vor, um die Ressourcen zu bereinigen, die Sie in diesem Tutorial erstellt haben:

- Falls Sie dies noch nicht getan haben, löschen Sie in [der S3-Konsole](#) alle Objekte und Buckets, die beim Ausführen der Anwendung erstellt wurden.

- Löschen Sie den Projektordner (getstarted).

## Nächste Schritte

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

- [Arbeiten mit Amazon S3](#)
- [Arbeiten mit anderen](#) Datenbankdiensten wie Amazon Web Services [DynamoDB](#) [Amazon EC2](#), und [verschiedenen Datenbankdiensten](#)
- [Verwenden Sie das SDK](#)
- [Sicherheit für AWS SDK for Java](#)

# Einrichten von AWS SDK for Java 2.x

Dieser Abschnitt enthält Informationen zum Einrichten Ihrer Entwicklungsumgebung und Projekte für die Verwendung von AWS SDK for Java 2.x.

## Übersicht über die Einrichtung

Für die erfolgreiche Entwicklung von Anwendungen, die AWS-Services über die auf zugreifen AWS SDK for Java, sind die folgenden Bedingungen erforderlich:

- Sie müssen [sich beim - AWS Zugriffsportal anmelden](#) können, das in der verfügbar ist AWS IAM Identity Center.
- Die [Berechtigungen der für das SDK konfigurierten IAM-Rolle](#) müssen den Zugriff auf die zulassen AWS-Services , die Ihre Anwendung benötigt. Die mit der PowerUserAccess AWS verwalteten Richtlinie verbundenen Berechtigungen reichen für die meisten Entwicklungsanforderungen aus.
- Eine Entwicklungsumgebung mit den folgenden Elementen:
  - [Freigegebene Konfigurationsdateien](#), die auf mindestens eine der folgenden Arten eingerichtet sind:
    - Die -configDatei enthält [IAM-Identity-Center-Single-Sign-On-Einstellungen](#), sodass das SDK AWS Anmeldeinformationen abrufen kann.
    - Die credentials Datei enthält temporäre Anmeldeinformationen.
  - Eine [Installation von Java 8](#) oder höher.
  - Ein [Build-Automatisierungstool](#) wie [Maven](#) oder [Gradle](#) .
  - Ein Texteditor für die Arbeit mit Code.
  - (Optional, aber empfohlen) Eine IDE (integrierte Entwicklungsumgebung) wie [IntelliJ IDEA](#), [Eclipse](#) oder [NetBeans](#).

Wenn Sie eine IDE verwenden, können Sie auch integrieren, AWS Toolkit um einfacher mit zu arbeiten AWS-Services. [AWS Toolkit for IntelliJ](#) und [AWS Toolkit for Eclipse](#) sind zwei Toolkits, die Sie für die Java-Entwicklung verwenden können.

- Eine aktive - AWS Zugriffsportalsitzung, wenn Sie bereit sind, Ihre Anwendung auszuführen. Sie verwenden die AWS Command Line Interface , um [den Anmeldevorgang beim Zugriffsportal von IAM Identity Center zu initiieren](#). AWS

### ⚠ Important

Die Anweisungen in diesem Einrichtungsabschnitt setzen voraus, dass Sie oder Organisation IAM Identity Center verwenden. Wenn Ihre Organisation einen externen Identitätsanbieter verwendet, der unabhängig von IAM Identity Center funktioniert, finden Sie heraus, wie Sie temporäre Anmeldeinformationen für das SDK for Java abrufen können. Befolgen Sie [diese Anweisungen](#), um der `~/.aws/credentials` Datei temporäre Anmeldeinformationen hinzuzufügen.

Wenn Ihr Identitätsanbieter der `~/.aws/credentials` Datei automatisch temporäre Anmeldeinformationen hinzufügt, stellen Sie sicher, dass der Profilname lautet, `[default]` damit Sie dem SDK oder keinen Profilnamen angeben müssen AWS CLI.

## Anmeldefähigkeit für das - AWS Zugriffsportal

Das - AWS Zugriffsportal ist der Speicherort, an dem 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 Anweisungen für den Kontozugriff im [IAM-Identity-Center-Authentifizierungsthema](#) im Referenzhandbuch für AWS SDKs und Tools.

## Einrichten des Single-Sign-On-Zugriffs für das SDK

Nachdem Sie Schritt 2 im [Abschnitt Programmgesteuerter 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-Zugriffsportalsitzung](#) starten, bevor Sie Ihre Anwendung ausführen.
- Eine `~/.aws/config` Datei, die ein [Standardprofil](#) enthält. Das SDK for Java verwendet die SSO-Token-Anbieterkonfiguration des Profils, um Anmeldeinformationen zu erhalten, bevor Anfragen an gesendet werden AWS. Der `sso_role_name` Wert, bei dem es sich um eine IAM-Rolle handelt, die mit einem IAM-Identity-Center-Berechtigungssatz verbunden ist, sollte den Zugriff auf die in Ihrer Anwendung AWS-Services verwendeten 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 -Zugriffssitzung.

```
[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 Einstellungen, die in der Konfiguration des SSO-Token-Anbieters verwendet werden, finden Sie unter [Konfiguration des SSO-Token-Anbieters](#) im Referenzhandbuch für AWS SDKs und Tools.

Wenn Ihre Entwicklungsumgebung nicht wie zuvor gezeigt für den programmgesteuerten Zugriff eingerichtet ist, befolgen Sie [Schritt 2 im Referenzhandbuch für -SDKs](#).

## Anmelden mit der AWS CLI

Bevor Sie eine Anwendung ausführen, die auf zugreift AWS-Services, benötigen Sie eine aktive - AWS Zugriffssitzung, damit das SDK die IAM-Identity-Center-Authentifizierung zum Auflösen von Anmeldeinformationen verwenden kann. Führen Sie den folgenden Befehl in der aus AWS CLI , um sich beim - AWS Zugriffportal anzumelden.

```
aws sso login
```

Da Sie ein Standardprofil eingerichtet haben, müssen Sie den Befehl nicht mit einer `--profile`-Option aufrufen. Wenn die Konfiguration Ihres SSO-Token-Anbieters 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 über eine aktive Sitzung verfügen.

```
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 Zugriffsportalansicht verfügen und ausführen `aws sso login`, müssen Sie keine Anmeldeinformationen angeben. Es wird jedoch ein Dialogfeld angezeigt, in dem die Berechtigung für den Zugriff `botocore` auf Ihre Informationen angefordert wird. `botocore` ist die Grundlage für AWS CLI. Wählen Sie Zulassen, um den Zugriff auf Ihre Informationen für die AWS CLI und SDK for Java zu autorisieren.

## Installieren von Java und eines Build-Tools

Ihre Entwicklungsumgebung benötigt Folgendes:

- Java 8 oder höher. Das AWS SDK for Java funktioniert mit dem [Oracle Java SE Development Kit](#) und mit Distributionen von Open Java Development Kit (OpenJDK), wie [Amazon Corretto](#), [Red Hat OpenJDK](#) und [Adoptium](#).
- 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 <https://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/>.

## Zusätzliche Authentifizierungsoptionen

Weitere Optionen zur Authentifizierung für das SDK, z. B. die Verwendung von Profilen und Umgebungsvariablen, finden Sie im [Konfigurationskapitel](#) im Referenzhandbuch für AWS SDKs und Tools.

# Richten Sie ein Apache Maven-Projekt ein

Sie können [Apache Maven](#) verwenden, um AWS SDK for Java Projekte einzurichten und zu erstellen oder um [das SDK selbst zu erstellen](#).

## Voraussetzungen

Um das AWS SDK for Java mit Maven zu verwenden, benötigen Sie Folgendes:

- Java 8.0 oder höher Sie können das neueste Java SE Development Kit unter <http://www.oracle.com/technetwork/java/javase/downloads/> herunterladen. Das AWS SDK for Java kann auch mit [OpenJDK](#) und Amazon Corretto, einer Verteilung des Open Java Development Kit (OpenJDK), verwendet werden. Laden Sie die neuste Version des OpenJDK unter <https://openjdk.java.net/install/index.html> herunter. Laden Sie die neueste Version Amazon Corretto 8 oder Amazon Corretto 11 von [der Corretto Seite](#) herunter.
- Apache Maven. Wenn Sie Maven installieren müssen, navigieren Sie zu <http://maven.apache.org/>, um das Tool herunterzuladen und zu installieren.

## Erstellen eines Maven-Projekts

Um ein Maven-Projekt von der Befehlszeile aus zu erstellen, führen Sie den folgenden Befehl in einem Terminal- oder Befehlszeilenfenster aus.

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

### Note

Ersetzen Sie `com.example.myapp` durch den vollständigen Paket-Namespaces Ihrer Anwendung. Ersetzen Sie auch `myapp` durch Ihren Projektnamen. Dies wird der Name des Verzeichnisses für Ihr Projekt.

[Um die neueste Version des Archetyps zu verwenden, ersetzen Sie 2.X.X durch die neueste Version von Maven Central.](#)

Dieser Befehl erstellt ein Maven-Projekt mit dem Archetype Templating Toolkit. Der Archetyp generiert das Gerüst für ein Function-Handler-Projekt. AWS Lambda Dieser Projekt-Archetyp ist für die Kompilierung mit Java SE 8 vorkonfiguriert und beinhaltet eine Abhängigkeit von der Version des SDK for Java 2.x, die mit spezifiziert ist. `-DarchetypeVersion`

Weitere Informationen zum Erstellen und Konfigurieren von Maven-Projekten finden Sie im [Maven-Handbuch „Erste Schritte“](#).

## Konfigurieren des Java-Compilers für Maven

Wenn Sie Ihr Projekt mit dem AWS Lambda Projekt-Archetyp wie zuvor beschrieben erstellt haben, ist die Konfiguration des Java-Compilers bereits für Sie erledigt.

Um zu überprüfen, ob diese Konfiguration vorhanden ist, öffnen Sie zunächst die Datei `pom.xml` aus dem Projektordner, den Sie erstellt haben (z. B. `myapp`), als Sie den vorherigen Befehl ausgeführt haben. In den Zeilen 11 und 12 sehen Sie die Einstellung der Java-Compiler-Version für dieses Maven-Projekt und in den Zeilen 71 bis 75 die erforderliche Einbindung des Maven-Compiler-Plugins.

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

Wenn Sie Ihr Projekt mit einem anderen Archetyp oder mit einer anderen Methode erstellen, müssen Sie sicherstellen, dass das Maven-Compiler-Plugin Teil des Builds ist und dass seine Quell- und Zieleigenschaften in der Datei beide auf 1.8 gesetzt sind. `pom.xml`

Eine Möglichkeit, diese erforderlichen Einstellungen zu konfigurieren, finden Sie im vorherigen Snippet.

Alternativ können Sie die Compiler-Konfiguration inline wie folgt mit der Plug-in-Deklaration konfigurieren.

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

## Deklarieren des SDK als Abhängigkeit

Sie können das AWS SDK for Java nur dann in Ihrem Projekt verwenden, wenn Sie es in der Datei `pom.xml` Ihres Projekts als Abhängigkeit deklarieren.

Wenn Sie Ihr Projekt mit dem Projekt-Archetyp wie zuvor beschrieben erstellt haben, ist die neueste Version des SDK bereits als Abhängigkeit in Ihrem Projekt konfiguriert.

Der Archetyp generiert eine Stücklisten-Artefaktabhängigkeit (Stückliste) für die Gruppen-ID `software.amazon.awssdk`. Bei einer Stückliste müssen Sie nicht die Maven-Version für einzelne Artefaktabhängigkeiten angeben, die dieselbe Gruppen-ID haben.

Wenn Sie Ihr Maven-Projekt auf andere Weise erstellt haben, konfigurieren Sie die neueste Version des SDK für Ihr Projekt, indem Sie sicherstellen, dass die Datei `pom.xml` Folgendes enthält.

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

```
<artifactId>bom</artifactId>
<version>${aws.java.sdk.version}</version>
<type>pom</type>
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
</project>
```

### Note

[Ersetzen Sie 2.X.X in der pom.xml Datei durch die neueste Version von AWS SDK for Java 2.x](#)

## Festlegen von Abhängigkeiten für SDK-Module

Nachdem Sie das SDK konfiguriert haben, können Sie Abhängigkeiten für eins oder mehrere der AWS SDK for Java-Module hinzufügen, die in Ihrem Projekt verwendet werden sollen.

Sie können zwar die Versionsnummer für jede Komponente angeben, müssen dies jedoch nicht, da Sie die SDK-Version bereits in dem dependencyManagement Abschnitt mit dem Stücklisten-Artefakt deklariert haben. Um eine andere Version eines bestimmten Moduls zu laden, geben Sie eine Versionsnummer für dessen Abhängigkeit an.

Wenn Sie Ihr Projekt mit dem Projekt-Archetyp wie zuvor beschrieben erstellt haben, ist Ihr Projekt bereits mit mehreren Abhängigkeiten konfiguriert. Dazu gehören Abhängigkeiten für AWS Lambda Funktionshandler und Amazon S3, wie folgt.

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

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

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

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

### Note

Im obigen pom.xml Beispiel stammen die Abhängigkeiten von verschiedenen s. groupId. Die s3 Abhängigkeit ist von software.amazon.awssdk, wohingegen die aws-lambda-java-core Abhängigkeit von ist com.amazonaws. Die Konfiguration der STL-Abhängigkeitsverwaltung wirkt sich auf Artefakte für aus software.amazon.awssdk, sodass für das aws-lambda-java-core Artefakt eine Version erforderlich ist. Für die Entwicklung von Lambda-Funktionshandlern, die das SDK for Java 2.x verwenden, aws-lambda-java-core ist dies die richtige Abhängigkeit. Wenn Ihre Anwendung jedoch Lambda-Ressourcen mithilfe von Operationen wie listFunctions, und createFunction verwalten muss deleteFunction invokeFunction, erfordert Ihre Anwendung die folgende Abhängigkeit.

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

### Note

Die s3 Abhängigkeit schließt die netty-nio-client und die apache-client transitiven Abhängigkeiten aus. Anstelle eines dieser HTTP-Clients beinhaltet der Archetyp die url-

connection-client Abhängigkeit, wodurch die [Startlatenz für Funktionen reduziert wird](#).  
AWS Lambda

Fügen Sie Ihrem Projekt die Module für die Funktionen hinzu, die AWS-Service Sie für Ihr Projekt benötigen. Die Module (Abhängigkeiten), die von der AWS SDK for Java BOM verwaltet werden, sind im [zentralen Maven-Repository](#) aufgeführt.

### Note

Sie können die Datei pom.xml anhand eines Codebeispiels betrachten, um zu ermitteln, welche Abhängigkeiten Sie für Ihr Projekt benötigen. Wenn Sie beispielsweise an den Abhängigkeiten für den DynamoDB-Dienst interessiert sind, sehen Sie sich [dieses Beispiel](#) aus dem [AWSCode Examples Repository](#) unter an. GitHub (Suchen Sie unter [/javav2/example\\_code/dynamodb](#) nach der pom.xml Datei.)

## Aufnehmen des gesamten SDK in Ihr Projekt

Zur Optimierung Ihrer Anwendung empfehlen wir dringend, nur die benötigten Komponenten abzurufen, und nicht das gesamte SDK. Wenn Sie jedoch das gesamte AWS SDK for Java in Ihr Projekt aufnehmen möchten, deklarieren Sie es wie folgt in Ihrer Datei pom.xml.

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

## Erstellen Ihres Projekts

Nachdem Sie die Datei pom.xml konfiguriert haben, können Sie Ihr Projekt mithilfe von Maven erstellen.

Um ein Maven-Projekt über die Befehlszeile zu erstellen, öffnen Sie ein Terminal- oder Eingabeaufforderungsfenster, navigieren Sie zu Ihrem Projektverzeichnis (z. B. myapp), geben oder fügen Sie den folgenden Befehl ein und drücken Sie dann die Eingabe.

```
mvn package
```

Dadurch wird eine einzelne `.jar`-Datei (JAR) im Verzeichnis `target` erstellt (z. B. myapp/target). Diese JAR enthält alle SDK-Module, die Sie als Abhängigkeiten in Ihrer Datei `pom.xml` angegeben haben.

## Richten Sie ein Gradle-Projekt ein

Sie können [Gradle](#) verwenden, um Projekte einzurichten und zu erstellen AWS SDK for Java.

Die ersten Schritte im folgenden Beispiel stammen aus dem [Handbuch Erste Schritte von Gradle](#) für Version 8.4. Wenn Sie eine andere Version verwenden, können Ihre Ergebnisse geringfügig abweichen.

Um eine Java-Anwendung mit Gradle (Befehlszeile) zu erstellen

1. Erstellen Sie ein Verzeichnis für Ihr Projekt. In diesem Beispiel demo ist das der Verzeichnisname.
2. Führen Sie innerhalb des demo Verzeichnisses den `gradle init` Befehl aus und geben Sie die rot markierten Werte ein, wie in der folgenden Befehlszeilenausgabe gezeigt. Für die Einführung wählen wir Kotlin als DSL-Sprache für das Build-Skript. Ein vollständiges Beispiel für Groovy finden Sie am Ende dieses Themas.

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

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

Select implementation language:
1: C++
2: Groovy
```



```
3: Java
4: Kotlin
5: Scala
6: Swift
Enter selection (default: Java) [1..6] 3

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

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

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

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

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

3. Nach Abschluss der `init` Aufgabe enthält das `demo` Verzeichnis die folgende Baumstruktur. Im nächsten Abschnitt schauen wir uns die Haupt-Build-Datei `build.gradle.kts` (rot hervorgehoben) genauer an.

```
### app
#   ### build.gradle.kts
#   ### src
#     ### main
#       #   ### java
#         #   #   ### demo
#           #   #   ### App.java
```

```
#      #   ### resources
#      ### test
#      ### java
#      #   ### demo
#      #       ### AppTest.java
#      ### resources
### gradle
#   ### wrapper
#       ### gradle-wrapper.jar
#       ### gradle-wrapper.properties
### gradlew
### gradlew.bat
### settings.gradle.kts
```

Die `build.gradle.kts` Datei enthält den folgenden gerüsteten Inhalt.

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

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

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

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

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

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

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

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

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

#### 4. Verwenden Sie die Gradle-Build-Datei mit Gerüsten als Grundlage für Ihr Projekt. AWS

- a. Um die SDK-Abhängigkeiten für Ihr Gradle-Projekt zu verwalten, fügen Sie dem Abschnitt der Datei die Maven-Stückliste (BOM) hinzu. AWS SDK for Java 2.x dependencies build.gradle.kts

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

#### Note

Ersetzen Sie in dieser Beispiel-Builddatei 2.21.1 durch die neueste Version des SDK for Java 2.x. Finden Sie die neueste Version, die im zentralen [Maven-Repository](#) verfügbar ist.

- b. Geben Sie im `dependencies` Abschnitt die SDK-Module an, die Ihre Anwendung benötigt. Im Folgenden wird beispielsweise eine Abhängigkeit von Amazon Simple Storage Service hinzugefügt.

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

Gradle löst automatisch die richtige Version der deklarierten Abhängigkeiten auf, indem es die Informationen aus der Stückliste verwendet.

Die folgenden Beispiele zeigen vollständige Gradle-Build-Dateien sowohl in den Kotlin- als auch in den Groovy-DSLs. Die Build-Datei enthält Abhängigkeiten für Amazon S3, Authentifizierung, Protokollierung und Tests. Die Quell- und Zielversion von Java ist Version 11.

Kotlin DSL (`build.gradle.kts`)

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

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

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

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

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

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

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

## Groovy DSL (build.gradle)

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

plugins {
```

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

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

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

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

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

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

Die nächsten Schritte finden Sie im Handbuch Erste Schritte auf der Gradle-Website mit Anweisungen zum [Erstellen und Ausführen einer Gradle-Anwendung](#).

# Richten Sie ein GraalVM Native Image-Projekt für AWS SDK for Java

Mit den Versionen 2.16.1 und höher AWS SDK for Java bietet das out-of-the-box Unterstützung für GraalVM Native Image-Anwendungen. Verwenden Sie den `archetype-app-quickstart` Maven-Archetyp, um ein Projekt mit integrierter nativer Image-Unterstützung einzurichten.

## Voraussetzungen

- Führen Sie die unter [Einrichtung der AWS SDK for Java 2.x](#) beschriebenen Schritte aus.
- Installieren Sie [GraalVM Native Image](#).

## Create a project using the archetype

Verwenden Sie den folgenden Befehl, um ein Maven-Projekt mit integrierter nativer Image-Unterstützung in einem Terminal- oder Befehlszeilenfenster zu erstellen.

### Note

`com.example.mynativeimageapp` Ersetzen Sie es durch den vollständigen Paket-Namespace Ihrer Anwendung. Ersetzen Sie es auch `mynativeimageapp` durch Ihren Projektnamen. Dies wird der Name des Verzeichnisses für Ihr Projekt.

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

Dieser Befehl erstellt ein Maven-Projekt, das mit Abhängigkeiten für den AWS SDK for Java Amazon S3, und den Apache HTTP-Client konfiguriert ist. Es enthält auch eine Abhängigkeit für das [GraalVM Native Image Maven-Plugin](#), sodass Sie native Images mit Maven erstellen können.

Um Abhängigkeiten für einen anderen Dienst einzubeziehen Amazon Web Services, setzen Sie den Wert des `Dservice` Parameters auf die Artefakt-ID dieses Dienstes. Beispiele hierfür sind `dynamodb`, `comprehend` und `pinpoint`. Eine vollständige Liste der Artefakt-IDs finden Sie in der Liste der verwalteten Abhängigkeiten für [software.amazon.awssdk auf Maven Central](#).

Um einen asynchronen HTTP-Client zu verwenden, setzen Sie den `DhttpClient` Parameter auf `netty-nio-client`. Um ihn stattdessen `URLConnectionHttpClient` als synchronen HTTP-Client zu verwenden `apache-client`, setzen Sie den `DhttpClient` Parameter auf `url-connection-client`.

## Erstellen Sie ein natives Image

Nachdem Sie das Projekt erstellt haben, führen Sie den folgenden Befehl in Ihrem Projektverzeichnis aus, z. B. `mynativeimageapp`:

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

Dadurch wird eine native Image-Anwendung im `target` Verzeichnis erstellt, zum Beispiel `target/mynativeimageapp`.



# Verwenden der AWS SDK for Java 2.x

Nachdem Sie die Schritte unter [Einrichten des SDK](#) abgeschlossen haben, können Sie Anforderungen an AWS Services wie Amazon S3, DynamoDB, IAM, Amazon EC2 und mehr stellen.

## Arbeiten mit Service-Clients

### Erstellen eines Service-Clients

Um eine Anfrage an einen zu stellen AWS-Service, müssen Sie zunächst einen Service-Client für diesen Service mithilfe der statischen Factory-Methode `instanciierenbuilder()`. Die `builder()` Methode gibt ein `builder` Objekt zurück, mit dem Sie den Service-Client anpassen können. Die praktischen Setter-Methoden geben das `builder`-Objekt zurück. So können Sie die Methodenaufrufe in Reihe schalten, was nicht nur einfacher ist, sondern auch für besser lesbaren Code sorgt. Nachdem Sie die gewünschten Eigenschaften konfiguriert haben, rufen Sie die `-build()` Methode auf, um den Client zu erstellen.

Der folgende Codeausschnitt instanziiert beispielsweise ein `Ec2Client` Objekt als Service-Client für Amazon EC2.

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

#### Note

Service-Clients im SDK sind threadsicher. Um eine optimale Leistung zu erzielen, behandeln Sie sie wie langlebige Objekte. Jeder Client verfügt über seine eigene Verbindungspool-Ressource, die freigegeben wird, wenn der Client von der Speicherbereinigung entfernt wird. Ein Service-Client-Objekt ist unveränderlich, daher müssen Sie für jeden Service, an den Sie Anforderungen stellen, einen neuen Client erstellen oder eine andere Konfiguration für Anforderungen an denselben Service verwenden.

Die Angabe von `Region` im Service-Client-Builder ist nicht für alle AWS Services erforderlich. Es hat sich jedoch bewährt, die Region für die API-Aufrufe festzulegen, die Sie in Ihren Anwendungen durchführen. Weitere Informationen finden Sie unter [AWS Auswahl der Region](#).

## Standard-Client-Konfiguration

Die Client-Generatoren haben eine weitere Factory-Methode mit dem Namen `create()`. Diese Methode erstellt einen Service-Client mit der Standard-Konfiguration. Es verwendet die Standard-Anbieterkette, um Anmeldeinformationen und zu laden AWS-Region. Wenn Anmeldeinformationen oder die Region nicht aus der Umgebung bestimmt werden können, in der die Anwendung ausgeführt wird, schlägt der Aufruf von `create` fehl. Weitere Informationen darüber, wie das SDK die zu verwendenden [Anmeldeinformationen](#) und [die zu verwendende Region bestimmt, finden Sie unter Verwenden von Anmeldeinformationen und Regionsauswahl](#).

Mit dem folgenden Codeausschnitt wird beispielsweise ein `DynamoDbClient` Objekt als Service-Client für Amazon DynamoDB instanziiert:

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

## Konfigurieren von Service-Clients

Um die Konfiguration eines Service-Clients anzupassen, verwenden Sie die Setter der `builder()` Factory-Methode. Verketteten Sie die Methoden, um mehrere Konfigurationsoptionen festzulegen und besser lesbaren Code zu erstellen.

Das folgende Beispiel zeigt ein `S3Client`, das mit mehreren benutzerdefinierten Einstellungen konfiguriert ist.

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

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

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

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

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

## Anfragen stellen

Verwenden Sie den Service-Client, um Anforderungen an das entsprechende zu stellen AWS-Service.

Dieser Codeausschnitt zeigt beispielsweise, wie Sie ein `RunInstancesRequest` Objekt erstellen, um eine neue Amazon EC2 zu erstellen:

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

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

Anstatt eine Anforderung zu erstellen und an die Instance zu übergeben, stellt das SDK Builder bereit, mit denen Sie eine Anforderung erstellen können. Mit einem Builder können Sie Java-Lambda-Ausdrücke verwenden, um die Anforderung „inline“ zu erstellen.

Im folgenden Beispiel wird das vorherige Beispiel neu geschrieben, indem die Version der `runInstances` [Methode verwendet wird, die einen Builder zum Erstellen der Anforderung verwendet](#).

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

## Antworten verarbeiten

Sie verwenden einen Antwort-Handler, um die Antwort von der zurückzuverarbeiten AWS-Service.

Dieser Codeausschnitt zeigt beispielsweise, wie Sie ein `-RunInstancesResponse` Objekt erstellen, um die Antwort von Amazon EC2 zu verarbeiten, indem Sie die `instanceId` für die neue Instance aus der obigen Anfrage drucken:

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

## Schließen des Service-Clients

Als bewährte Methode sollten Sie einen Service-Client für mehrere API-Service-Aufrufe während der Lebensdauer einer Anwendung verwenden. Wenn Sie jedoch einen Service-Client für eine einmalige Verwendung benötigen oder den Service-Client nicht mehr benötigen, schließen Sie ihn.

Rufen Sie die `-close()` Methode auf, wenn der Service-Client nicht mehr benötigt wird, um Ressourcen freizugeben.

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

Wenn Sie einen Service-Client für die einmalige Verwendung benötigen, können Sie den Service-Client als Ressource in einer `try-with-resources`-Anweisung instanziiieren. Service-Clients implementieren die [-Autoclosable](#) Schnittstelle, sodass das JDK die `close()` Methode am Ende der Anweisung automatisch aufruft.

Das folgende Beispiel zeigt, wie Sie einen Service-Client für einen einmaligen Aufruf verwenden. Das `StsClient`, das aufruft, AWS Security Token Service wird geschlossen, nachdem es die Konto-ID zurückgegeben hat.

```
import software.amazon.awssdk.services.sts.StsClient;  
  
String getAccountID() {  
    try (StsClient stsClient = StsClient.create()) {  
        return stsClient.getCallerIdentity().account();  
    }  
}
```

## Ausnahmen behandeln

Das SDK verwendet Laufzeitausnahmen (oder nicht überprüfte Ausnahmen), um Ihnen eine detaillierte Kontrolle über die Fehlerbehandlung zu ermöglichen und sicherzustellen, dass die Ausnahmebehandlung mit Ihrer Anwendung skaliert wird.

Ein [SdkServiceException](#) oder eine seiner Unterklassen ist die gängigste Form der Ausnahme, die das SDK auslöst. Diese Ausnahmen stellen Antworten vom AWS Service dar. Sie können auch eine [SdkClientException](#) behandeln, die auftritt, wenn clientseitig ein Problem auftritt (d. h. in Ihrer Entwicklungs- oder Anwendungsumgebung), z. B. ein Netzwerkverbindungsfehler.

Dieser Codeausschnitt zeigt eine Möglichkeit, Serviceausnahmen zu behandeln, wenn Sie eine Datei in hochladen Amazon S3. Der Beispielcode fängt sowohl Client- als auch Serverausnahmen ab, protokolliert die Details und existiert in der Anwendung.

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

try {

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

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

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

Weitere Informationen finden Sie unter [Umgang mit Ausnahmen](#).

## Verwenden von Waitern

Die Verarbeitung einiger Anfragen dauert einige Zeit, z. B. das Erstellen einer neuen Tabelle in DynamoDB oder das Erstellen eines neuen Amazon S3 Buckets. Um sicherzustellen, dass die Ressource bereit ist, bevor Ihr Code weiter ausgeführt wird, verwenden Sie einen Waiter .

Dieser Codeausschnitt erstellt beispielsweise eine neue Tabelle („myTable“) in DynamoDB, wartet, bis sich die Tabelle im ACTIVE Status befindet, und gibt dann die Antwort aus:

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

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

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

Weitere Informationen finden Sie unter [Verwenden von Waitern](#).

## HTTP-Clients

Sie können die Standardkonfiguration für HTTP-Clients in Anwendungen ändern, die Sie mit der erstellen AWS SDK for Java. Informationen zum Konfigurieren von HTTP-Clients und -Einstellungen finden Sie unter [HTTP-Konfiguration](#) .

## Wiederholversuche

Sie können die Standardeinstellungen für Wiederholungen in Ihren Service-Clients ändern, einschließlich des Wiederholungsmodus und der Back-off-Strategie. Weitere Informationen finden Sie in [der -RetryPolicyKlasse](#) in der AWS SDK for Java -API-Referenz.

Weitere Informationen zu Wiederholungsversuchen in - AWS Services finden Sie unter [Wiederholversuche bei Fehlern und exponentielles Backoff in AWS](#).

## Timeouts

Sie können Timeouts für jeden Ihrer Service-Clients mithilfe der `apiCallTimeout` und der `apiCallAttemptTimeout` Setter konfigurieren. Die `apiCallTimeout` Einstellung ist

die Zeit, die der Client benötigt, um die Ausführung eines API-Aufrufs abzuschließen. Die `apiCallAttemptTimeout` Einstellung ist die Zeit, die gewartet werden muss, bis die HTTP-Anforderung abgeschlossen ist, bevor sie aufgibt.

Weitere Informationen finden Sie unter [apiCallTimeout](#) und [apiCallAttemptTimeout](#) in der AWS SDK for Java -API-Referenz.

## Ausführungsabfänger

Sie können Code schreiben, der die Ausführung Ihrer API-Anforderungen und -Antworten an verschiedenen Teilen des Anforderungs-/Antwortlebenszyklus abfängt. Auf diese Weise können Sie Metriken veröffentlichen, eine laufende Anfrage ändern, die Anfrageverarbeitung debuggen, Ausnahmen anzeigen und vieles mehr. Weitere Informationen finden Sie in [der -ExecutionInterceptorSchnittstelle](#) in der AWS SDK for Java -API-Referenz.

## Zusätzliche Informationen

- Vollständige Beispiele für die obigen Codeausschnitte finden Sie unter [Arbeiten mit Amazon DynamoDB](#), [Arbeiten mit Amazon EC2](#) und [Arbeiten mit Amazon S3](#).

## Bereitstellen temporärer Anmeldeinformationen für das SDK

Bevor Sie eine Anfrage an Amazon Web Services über die `stellen` AWS SDK for Java 2.x, signiert das SDK kryptografisch temporäre Anmeldeinformationen, die von `ausgestellt` wurden AWS. Um auf temporäre Anmeldeinformationen zuzugreifen, ruft das SDK Konfigurationswerte ab, indem es mehrere Speicherorte überprüft.

In diesem Thema werden verschiedene Möglichkeiten beschrieben, wie Sie dem SDK den Zugriff auf temporäre Anmeldeinformationen ermöglichen.

### Themen

- [Konfigurieren des Zugriffs auf temporäre Anmeldeinformationen](#)
- [Standard-Anbieterkette für Anmeldeinformationen](#)
- [Verwenden eines bestimmten Anbieters von Anmeldeinformationen oder einer bestimmten Anbieterkette](#)
- [Verwenden von Profilen](#)

- [Laden temporärer Anmeldeinformationen aus einem externen Prozess](#)
- [Temporäre Anmeldeinformationen im Code bereitstellen](#)
- [Lesen von IAM-Rollenanmeldeinformationen auf Amazon EC2](#)

## Konfigurieren des Zugriffs auf temporäre Anmeldeinformationen

Für mehr Sicherheit AWS empfiehlt, das SDK für Java so zu konfigurieren, dass [temporäre Anmeldeinformationen anstelle langlebiger Anmeldeinformationen verwendet](#) werden. Temporäre Anmeldeinformationen bestehen aus Zugriffsschlüsseln (Zugriffsschlüssel-ID und geheimer Zugriffsschlüssel) und einem Sitzungstoken. Wir empfehlen, [das SDK so zu konfigurieren](#), dass temporäre Anmeldeinformationen automatisch abgerufen werden, da der Token-Aktualisierungsprozess automatisch erfolgt. Sie können [dem SDK jedoch direkt temporäre Anmeldeinformationen bereitstellen](#).

### IAM-Identity-Center-Konfiguration

Wenn Sie das SDK für die Verwendung des Single-Sign-On-Zugriffs von IAM Identity Center konfigurieren, wie [???](#) in diesem Handbuch beschrieben, verwendet das SDK automatisch temporäre Anmeldeinformationen.

Das SDK verwendet das IAM Identity Center-Zugriffstoken, um Zugriff auf die IAM-Rolle zu erhalten, die mit der `-sso_role_name`-Einstellung in Ihrer `-configDatei` konfiguriert ist. Das SDK übernimmt diese IAM-Rolle und ruft temporäre Anmeldeinformationen ab, die für `-AWS-ServiceAnforderungen` verwendet werden sollen.

Weitere Informationen darüber, wie das SDK temporäre Anmeldeinformationen aus der Konfiguration erhält, finden Sie im Abschnitt [Grundlegendes zur IAM-Identity-Center-Authentifizierung](#) im Referenzhandbuch für AWS SDKs und Tools.

### Abrufen aus dem -AWSZugriffsportale

Als Alternative zur Single-Sign-On-Konfiguration von IAM Identity Center können Sie temporäre Anmeldeinformationen kopieren und verwenden, die im `-AWSZugriffsportale` verfügbar sind. Sie können die temporären Anmeldeinformationen in einem Profil verwenden oder sie als Werte für Systemeigenschaften und Umgebungsvariablen verwenden.

Einrichten einer lokalen Anmeldeinformationsdatei für temporäre Anmeldeinformationen

1. [Erstellen einer freigegebenen Anmeldeinformationsdatei](#)





## Standard-Anbieterkette für Anmeldeinformationen

Die standardmäßige Anbieterkette für Anmeldeinformationen wird von der [-DefaultCredentialsProvider](#)-Klasse implementiert. Es überprüft nacheinander jeden Ort, an dem Sie die Standardkonfiguration für die Bereitstellung temporärer Anmeldeinformationen festlegen können, und wählt dann die erste aus, die Sie festlegen.

Um die standardmäßige Anmeldeinformationsanbieterkette zur Bereitstellung temporärer Anmeldeinformationen zu verwenden, die in Ihrer Anwendung verwendet werden, erstellen Sie einen Service-Client-Builder, geben Sie jedoch keinen Anmeldeinformationsanbieter an. Der folgende Codeausschnitt erstellt einen `DynamoDbEnhancedClient`, der die standardmäßige Anmeldeinformationsanbieterkette verwendet, um Standardkonfigurationseinstellungen zu finden und abzurufen.

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

## Reihenfolge des Abrufs von Anmeldeinformationseinstellungen

Die standardmäßige Anmeldeinformationsanbieterkette des SDK for Java 2.x sucht nach der Konfiguration in Ihrer Umgebung mit einer vordefinierten Sequenz.

### 1. Java-Systemeigenschaften

- Das SDK verwendet die [SystemPropertyCredentialsProvider](#)-Klasse `aws.accessKeyId`, um temporäre Anmeldeinformationen aus den `aws.sessionToken` Java-Systemeigenschaften `aws.secretAccessKey`, und zu laden.

#### Note

Informationen zum Festlegen von Java-Systemeigenschaften finden Sie im Tutorial [Systemeigenschaften](#) auf der offiziellen Java-Tutorials-Website.

### 2. Umgebungsvariablen

- Das SDK verwendet die [EnvironmentVariableCredentialsProvider](#)-Klasse `AWS_ACCESS_KEY_ID`, um temporäre Anmeldeinformationen aus den `AWS_SESSION_TOKEN` Umgebungsvariablen `AWS_SECRET_ACCESS_KEY`, und zu laden.

### 3. Web-Identitätstoken von AWS Security Token Service

- Das SDK verwendet die [WebIdentityTokenFileCredentialsProvider](#) Klasse , um temporäre Anmeldeinformationen aus Java-Systemeigenschaften oder Umgebungsvariablen zu laden.

### 4. Die config freigegebenen Dateien credentials und

- Das SDK verwendet die [ProfileCredentialsProvider](#), um IAM-Identity-Center-Single-Sign-On-Einstellungen oder temporäre Anmeldeinformationen aus dem [default] Profil in den freigegebenen config Dateien credentials und zu laden.

Das Referenzhandbuch zu AWS SDKs und Tools enthält [detaillierte Informationen](#) darüber, wie das SDK for Java mit dem Single-Sign-On-Token von IAM Identity Center funktioniert, um temporäre Anmeldeinformationen abzurufen, die das SDK zum Aufrufen von verwendetAWS-Services.

#### Note

Die config Dateien credentials und werden von verschiedenen AWS SDKs und Tools gemeinsam genutzt. Weitere Informationen finden Sie unter [Die .aws/credentials- und .aws/config-Dateien](#) im Referenzhandbuch für AWS SDKs und Tools.

### 5. Amazon ECS -Container-Anmeldeinformationen

- Das SDK verwendet die [ContainerCredentialsProvider](#) Klasse , um temporäre Anmeldeinformationen aus der AWS\_CONTAINER\_CREDENTIALS\_RELATIVE\_URI Systemumgebungsvariablen zu laden.

### 6. Amazon EC2 Von der -Instance-IAM-Rolle bereitgestellte Anmeldeinformationen

- Das SDK verwendet die [InstanceProfileCredentialsProvider](#) Klasse , um temporäre Anmeldeinformationen aus dem Amazon EC2 Metadaten-Service zu laden.

## Verwenden eines bestimmten Anbieters von Anmeldeinformationen oder einer bestimmten Anbieterkette

Als Alternative zur standardmäßigen Anbieterkette von Anmeldeinformationen können Sie angeben, welchen Anbieter von Anmeldeinformationen das SDK verwenden soll. Wenn Sie einen bestimmten Anmeldeinformationsanbieter angeben, überspringt das SDK die Überprüfung verschiedener Standorte, wodurch die Zeit zum Erstellen eines Service-Clients etwas reduziert wird.

Wenn Sie beispielsweise Ihre Standardkonfiguration mithilfe von Umgebungsvariablen festlegen, stellen Sie der `-credentialsProvider` Methode im Service-Client-Builder ein [EnvironmentVariableCredentialsProvider](#) Objekt bereit, wie im folgenden Codeausschnitt.

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

Eine vollständige Liste der Anbieter von Anmeldeinformationen und Anbieterketten finden Sie unter [Alle bekannten Implementierungsklassen in AwsCredentialsProvider](#).

### Note

Sie können Ihren eigenen Anmeldeinformationsanbieter oder Ihre eigenen Anbieterketten verwenden, indem Sie die `-AwsCredentialsProvider` Schnittstelle implementieren.

## Verwenden von Profilen

Mit der freigegebenen `-config` und `-credentials` Datei können Sie mehrere Profile einrichten. Auf diese Weise kann Ihre Anwendung mehrere Gruppen von Anmeldeinformationen verwenden. Das `[default]` Profil wurde zuvor erwähnt. Das SDK verwendet die [ProfileCredentialsProvider](#) Klasse, um Einstellungen aus Profilen zu laden, die in der freigegebenen `credentials` Datei definiert sind.

Der folgende Codeausschnitt zeigt, wie Sie einen Service-Client erstellen, der die Einstellungen verwendet, die als Teil des Profils mit dem Namen definiert sind `my_profile`.

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

## Ein anderes Profil als Standard festlegen

Um ein anderes Profil als das `[default]` Profil als Standard für Ihre Anwendung festzulegen, legen Sie die `AWS_PROFILE` Umgebungsvariable auf den Namen Ihres benutzerdefinierten Profils fest.

Um diese Variable unter Linux, macOS oder Unix festzulegen, verwenden Sie `export`:

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

In Windows können Sie die Variablen mit `set` festlegen:

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

Alternativ können Sie die `aws.profile` Java-Systemeigenschaft auf den Namen des Profils setzen.

## Profilanmeldeinformationen neu laden

Sie können jeden Anmeldeinformationsanbieter konfigurieren, der über eine `profileFile()` Methode in seinem Builder verfügt, um Profilanmeldeinformationen neu zu laden.

Diese Anmeldeinformationsprofilklassen sind: `ProfileCredentialsProvider`, `DefaultCredentialsProviderInstanceProfileCredentialsProvider`, und `ProfileTokenProvider`.

### Note

Das erneute Laden von Profilanmeldeinformationen funktioniert nur mit den folgenden Einstellungen in der Profildatei: `aws_access_key_id`, `aws_secret_access_key`, und `aws_session_token`.

Einstellungen wie `region`, `sso_session`, `sso_account_id`, und `source_profile` werden ignoriert.

Um einen unterstützten Anmeldeinformationsanbieter zum Neuladen von Profileinstellungen [ProfileFileSupplier](#) zu konfigurieren, geben Sie eine Instance von für die `profileFile()` Builder-Methode an. Das folgende Codebeispiel zeigt eine `ProfileCredentialsProvider`, die Einstellungen für Anmeldeinformationen aus dem [default] Profil neu lädt.

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

// Set up a service client with the provider instance.
DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
```

```

        .region(Region.US_EAST_1)
        .credentialsProvider(provider)
        .build();

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

```

Wenn aufgerufen `ProfileCredentialsProvider.resolveCredentials()` wird, lädt das SDK for Java die Einstellungen neu. `ProfileFileSupplier.defaultSupplier()` ist eine von [mehreren Implementierungen](#) von `ProfileFileSupplier`, die vom SDK `ProfileFileSupplier` bereitgestellt werden. Wenn Ihr Anwendungsfall dies erfordert, können Sie Ihre eigene Implementierung bereitstellen.

Das folgende Beispiel zeigt die Verwendung der `ProfileFileSupplier.reloadWhenModified()` Convenience-Methode. `reloadWhenModified()` verwendet einen Path Parameter, der Ihnen Flexibilität bei der Angabe der Quelldatei für die Konfiguration anstelle des Standardspeicherorts `~/.aws/credentials` (oder `config`) gibt.

Die Einstellungen werden neu geladen, wenn nur aufgerufen `resolveCredentials()` wird, wenn SDK feststellt, dass der Inhalt der Datei geändert wurde.

```

Path credentialsFilePath = ...

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

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

```

Die `ProfileFileSupplier.aggregate()` Methode führt den Inhalt mehrerer Konfigurationsdateien zusammen. Sie entscheiden, ob eine Datei pro Aufruf neu geladen wird `resolveCredentials()` oder ob die Einstellungen einer Datei zum Zeitpunkt des ersten Lesens festgelegt wurden.

Das folgende Beispiel zeigt eine `DefaultCredentialsProvider`, die die Einstellungen von zwei Dateien zusammenführt, die Profileinstellungen enthalten. Das SDK lädt die Einstellungen in der Datei, auf die die `credentialsFilePath` Variable verweist, jedes Mal neu, wenn aufgerufen `resolveCredentials()` wird und sich die Einstellungen geändert haben. Die Einstellungen des `profileFile` Objekts bleiben unverändert.

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

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

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

## Laden temporärer Anmeldeinformationen aus einem externen Prozess

### Warning

Im Folgenden wird eine Methode beschrieben, bei der temporäre Anmeldeinformationen aus einem externen Prozess bezogen werden. Dies kann potenziell gefährlich sein. Seien Sie daher vorsichtig. Andere Anbieter von Anmeldeinformationen sollten nach Möglichkeit bevorzugt werden. Wenn Sie diese Option verwenden, sollten Sie sicherstellen, dass die `config` Datei mithilfe der bewährten Sicherheitsmethoden für Ihr Betriebssystem so gesperrt wie möglich ist.

Stellen Sie sicher, dass Ihr Tool für benutzerdefinierte Anmeldeinformationen keine geheimen Informationen in `writeStdErr`. -SDKs und AWS CLI können solche Informationen erfassen und protokollieren und sie möglicherweise nicht autorisierten Benutzern zugänglich machen.

Mit dem SDK for Java 2.x können Sie temporäre Anmeldeinformationen von einem externen Prozess für benutzerdefinierte Anwendungsfälle erwerben. Es gibt zwei Möglichkeiten, diese Funktionalität zu konfigurieren.

## Verwenden der `-credential_process` Einstellung

Wenn Sie über eine Methode verfügen, die temporäre Anmeldeinformationen bereitstellt, können Sie diese integrieren, indem Sie die `credential_process` Einstellung als Teil einer Profildefinition in der `config` Datei hinzufügen. Der von Ihnen angegebene Wert muss den vollständigen Pfad zur Befehlsdatei verwenden. Wenn der Dateipfad Leerzeichen enthält, müssen Sie ihn mit Anführungszeichen umgeben.

Das SDK ruft den Befehl genau wie angegeben auf und liest dann JSON-Daten aus `stdout`.

Die folgenden Beispiele zeigen die Verwendung dieser Einstellung für Dateipfade ohne Leerzeichen und Dateipfade mit Leerzeichen.

### Linux/macOS

#### Keine Leerzeichen im Dateipfad

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

#### Bereiche im Dateipfad

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

### Windows

#### Keine Leerzeichen im Dateipfad

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



## Bereiche im Dateipfad

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

Der folgende Codeausschnitt zeigt, wie Sie einen Service-Client erstellen, der die temporären Anmeldeinformationen verwendet, die als Teil des Profils mit dem Namen definiert sind `process-credential-profile`.

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

Ausführliche Informationen zur Verwendung eines externen Prozesses als Quelle temporärer Anmeldeinformationen finden Sie im [Abschnitt Prozessanmeldeinformationen](#) im Referenzhandbuch für AWS SDKs und Tools.

## Verwenden der **ProcessCredentialsProvider**

Alternativ zur Verwendung von Einstellungen in der config Datei können Sie die des SDK verwenden, [ProcessCredentialsProvider](#) um temporäre Anmeldeinformationen mit Java zu laden.

Die folgenden Beispiele zeigen verschiedene Versionen von , wie Sie einen externen Prozess mithilfe der angeben `ProcessCredentialsProvider` und einen Service-Client konfigurieren, der die temporären Anmeldeinformationen verwendet.

### Linux/macOS

#### Keine Leerzeichen im Dateipfad

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

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

## Bereiche im Dateipfad

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

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

## Windows

### Keine Leerzeichen im Dateipfad

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

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

### Bereiche im Dateipfad

```
ProcessCredentialsProvider credentials =
    ProcessCredentialsProvider
        .builder()
        .command("\"C:\\Path\\With Spaces To\\credentials.exe\" optional_param1
optional_param2")
```

```
        .build();

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

## Temporäre Anmeldeinformationen im Code bereitstellen

Wenn die Standard-Anmeldeinformationskette oder ein bestimmter oder benutzerdefinierter Anbieter oder eine Anbieterkette nicht für Ihre Anwendung funktioniert, können Sie temporäre Anmeldeinformationen direkt im Code bereitstellen. Dabei kann es sich um [Anmeldeinformationen für IAM-Rollen](#) handeln, wie [oben beschrieben](#), oder um temporäre Anmeldeinformationen, die von AWS Security Token Service (AWS STS) abgerufen wurden. Wenn Sie temporäre Anmeldeinformationen mit abgerufen haben AWS STS, stellen Sie sie einem `-AWS-ServiceClient` bereit, wie im folgenden Codebeispiel gezeigt.

1. Übernehmen Sie eine Rolle, indem Sie aufrufen `StsClient.assumeRole()`.
2. Erstellen Sie ein [StaticCredentialsProvider](#) Objekt und stellen Sie es mit dem `AwsSessionCredentials` Objekt bereit.
3. Konfigurieren Sie den Service-Client-Builder mit und `StaticCredentialsProvider` erstellen Sie den Client.

Im folgenden Beispiel wird ein Amazon S3-Serviceclient mit temporären Anmeldeinformationen erstellt, die von AWS STS für eine IAM-angenommene Rolle zurückgegeben werden.

```
// The AWS IAM Identity Center identity (user) who executes this method does not
// have permission to list buckets.
// The identity is configured in the [default] profile.
public static void assumeRole(String roleArn, String roleSessionName) {
    // The IAM role represented by the 'roleArn' parameter can be assumed by
    // identities in two different accounts
    // and the role permits the user to only list buckets.

    // The SDK's default credentials provider chain will find the single sign-on
    // settings in the [default] profile.
    // The identity configured with the [default] profile needs permission to call
    // AssumeRole on the STS service.
    try {
```

```

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

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

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

```

## Berechtigungssatz

Der folgende Berechtigungssatz, der in definiert ist, AWS IAM Identity Center ermöglicht es der Identität (Benutzer), die folgenden beiden Operationen auszuführen

1. Der GetObject Betrieb des Amazon Simple Storage Service.
2. Die -AssumeRoleOperation des AWS Security Token Service.

Ohne Übernahme der Rolle würde die im Beispiel gezeigte `s3.listBuckets()` Methode fehlschlagen.

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

## Angenommene Rolle

### Richtlinie für übernommene Rollenberechtigungen

Die folgende Berechtigungsrichtlinie ist an die Rolle angehängt, die im vorherigen Beispiel übernommen wurde. Diese Berechtigungsrichtlinie ermöglicht die Möglichkeit, alle Buckets im selben Konto wie die Rolle aufzulisten.

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

## Vertrauensrichtlinie für übernommene Rollen

Die folgende Vertrauensrichtlinie ist an die Rolle angehängt, die im vorherigen Beispiel übernommen wurde. Die Richtlinie ermöglicht die Übernahme der Rolle durch Identitäten (Benutzer) in zwei Konten.

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

## Lesen von IAM-Rollenanmeldeinformationen auf Amazon EC2

Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2-Instance ausgeführt werden und - AWS CLI oder AWS -API-Anforderungen stellen. Das ist eher zu empfehlen, als Zugriffsschlüssel innerhalb der EC2-Instance zu speichern. Um einer EC2-Instance eine - AWS Rolle zuzuweisen und sie für alle ihre Anwendungen verfügbar zu machen, erstellen Sie ein Instance-Profil, das der Instance zugeordnet ist. Ein Instance-Profil enthält die Rolle und ermöglicht, dass Programme, die in der EC2-Instance ausgeführt werden, temporäre Anmeldeinformationen erhalten. Weitere Informationen finden Sie unter [Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden](#) im IAM-Benutzerhandbuch.

Dieses Thema enthält Informationen zum Einrichten Ihrer Java-Anwendung für die Ausführung auf einer EC2-Instance und zum Aktivieren des SDK for Java zum Abrufen von IAM Rollenanmeldeinformationen.

## Anfordern von IAM-Rollenanmeldeinformationen aus der Umgebung

Wenn Ihre Anwendung mithilfe der `-create`-Methode (oder `-builder().build()`-Methoden) einen `- AWS Service-Client` erstellt, verwendet das SDK for Java die standardmäßige Anbieterkette für Anmeldeinformationen. Die standardmäßige Anmeldeinformationsanbieterkette durchsucht die Ausführungsumgebung nach Konfigurationselementen, die das SDK gegen temporäre Anmeldeinformationen eintauschen kann. Der [the section called “Standard-Anbieterkette für Anmeldeinformationen”](#) Abschnitt beschreibt den vollständigen Suchvorgang.

Der letzte Schritt in der Standardanbieterkette ist nur verfügbar, wenn Ihre Anwendung auf einer Instance ausgeführt wird Amazon EC2 . In diesem Schritt verwendet das SDK eine `InstanceProfileCredentialsProvider` um die im EC2-Instance-Profil definierte IAM-Rolle zu lesen. Das SDK erwirbt dann temporäre Anmeldeinformationen für diese IAM-Rolle.

Obwohl diese Anmeldeinformationen temporär sind und irgendwann ablaufen würden, aktualisiert ein `InstanceProfileCredentialsProvider` regelmäßig für Sie, sodass sie weiterhin Zugriff auf gewähren AWS.

## Programmgesteuertes Erfassen von Anmeldeinformationen für IAM-Rollen

Als Alternative zur standardmäßigen Anbieterkette für Anmeldeinformationen, die schließlich einen `InstanceProfileCredentialsProvider` auf EC2 verwendet, können Sie einen Service-Client explizit mit einem konfigurieren `InstanceProfileCredentialsProvider`. Dieser Ansatz wird im folgenden Ausschnitt gezeigt.

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

## Sicheres Abrufen von Anmeldeinformationen für IAM-Rollen

Standardmäßig führen EC2-Instances [IMDS](#) (Instance Metadata Service) aus, mit dem die SDKs `InstanceProfileCredentialsProvider` auf Informationen wie die konfigurierte IAM-Rolle zugreifen können. EC2-Instances führen standardmäßig zwei Versionen von IMDS aus:

- Instance-Metadatenservice Version 1 (IMDSv1) – Ein Anfrage/Antwort-Verfahren
- Instance-Metadatenservice Version 2 (IMDSv2) – Ein sitzungorientiertes Verfahren

[IMDSv2 ist ein sichererer Ansatz](#) als IMDSv1.

Standardmäßig versucht das Java-SDK zunächst IMDSv2, die IAM-Rolle abzurufen, aber wenn dies fehlschlägt, versucht es IMDSv1. Da IMDSv1 jedoch weniger sicher ist, AWS empfiehlt nur die Verwendung von IMDSv2 und , um das SDK daran zu hindern, IMDSv1 auszuprobieren.

Um den sichereren Ansatz zu verwenden, deaktivieren Sie die Verwendung von IMDSv1 durch das SDK, indem Sie eine der folgenden Einstellungen mit dem Wert bereitstellen `true`.

- Umgebungsvariable: `AWS_EC2_METADATA_V1_DISABLED`
- JVM-Systemeigenschaft: `aws.disableEc2MetadataV1`
- Freigegebene Konfigurationsdateieinstellung: `ec2_metadata_v1_disabled`

Wenn eine dieser Einstellungen auf `true` gesetzt ist, lädt das SDK keine IMDS-Rollenanmeldeinformationen mithilfe von IMDSv1, wenn der erste IMDSv2-Aufruf fehlschlägt.

## Verwenden von AWS-Regionen

AWS-Regionen ermöglichen es Service-Clients, auf zuzugreifen AWS-Services , die sich physisch in einem bestimmten geografischen Gebiet befinden.

### Explizite Konfiguration eines AWS-Region

Um eine Region explizit festzulegen, empfehlen wir Ihnen, die in der [Region](#) sklassedefinierten Konstanten zu verwenden. Dabei handelt es sich um eine Aufzählung aller öffentlich verfügbaren Regionen.

Um einen Client mit einer aufgezählten Region aus der Klasse zu erstellen, verwenden Sie die `region` Methode des Client Builders.

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

Wenn die Region, die Sie verwenden möchten, keine der Aufzählungen in der Klasse `Region` ist, können Sie mithilfe der statischen `of` Methode eine neue Region erstellen. Diese Methode ermöglicht Ihnen den Zugriff auf neue Regionen, ohne das SDK zu aktualisieren.

```
Region newRegion = Region.of("us-east-42");
Ec2Client ec2 = Ec2Client.builder()
    .region(newRegion)
```



```
.build();
```

### Note

Nachdem Sie einen Client mit dem Builder erstellt haben, ist er unveränderlich und kann AWS-Region nicht mehr geändert werden. Wenn Sie mit mehreren AWS-Regionen für denselben Service arbeiten müssen, sollten Sie mehrere Clients erstellen – einen pro Region.

## Lassen Sie das SDK die Region automatisch aus der Umgebung ermitteln

Wenn Ihr Code auf Amazon EC2 oder ausgeführt wird AWS Lambda, sollten Sie Clients so konfigurieren, dass sie dieselbe verwenden AWS-Region, auf der Ihr Code ausgeführt wird. Dies entkoppelt Ihren Code von der Umgebung, in der er ausgeführt wird, und erleichtert die Bereitstellung Ihrer Anwendung in mehreren AWS-Regionen für eine geringere Latenz oder Redundanz.

Um die standardmäßige Kette von Anmeldeinformationen/Regionsanbietern zu verwenden, um die Region aus der Umgebung zu bestimmen, verwenden Sie die `create` Methode des Client-Builders.

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

Wenn Sie ein nicht explizit AWS-Region mithilfe der `-region` Methode festlegen, konsultiert das SDK die standardmäßige Regionsanbieterkette, um die zu verwendende Region zu bestimmen.

## Verstehen der standardmäßigen Regionsanbieterkette

Das SDK führt die folgenden Schritte aus, um nach einem AWS-Region zu suchen:

1. Jede explizite Region, die mithilfe von `region` auf dem Builder selbst festgelegt wird, hat Vorrang vor allem anderen.
2. Die Umgebungsvariable `AWS_REGION` wird geprüft. Wenn sie festgelegt ist, wird diese Region zur Konfiguration des Clients verwendet.

### Note

Der Lambda Container legt diese Umgebungsvariable fest.

3. Das SDK überprüft die AWS freigegebene Konfigurationsdatei und die freigegebene Anmeldeinformationsdatei (in der Regel unter `~/.aws/config` und `~/.aws/credentials`). Wenn die `region` Eigenschaft vorhanden ist, verwendet das SDK sie.
  - Wenn das SDK die `-region` Eigenschaft in beiden Dateien für dasselbe Profil (einschließlich des `-default` Profils) findet, verwendet das SDK den Wert in der Datei mit den gemeinsam genutzten Anmeldeinformationen.
  - Die Umgebungsvariable `AWS_CONFIG_FILE` kann verwendet werden, um den Speicherort der gemeinsam genutzten Konfigurationsdatei anzupassen.
  - Die `AWS_PROFILE` Umgebungsvariable oder die `aws.profile` Systemeigenschaft kann verwendet werden, um das Profil anzugeben, das das SDK lädt.
4. Das SDK versucht, den Amazon EC2 Instance Metadata Service (IMDS) zu verwenden, um die Region der aktuell Amazon EC2 ausgeführten Instance zu bestimmen.
  - Um die Sicherheit zu erhöhen, sollten Sie das SDK daran hindern, Version 1 von IMDS zu verwenden. Sie verwenden dieselbe Einstellung, um Version 1 zu deaktivieren, die im [the section called "Sicheres"](#) Abschnitt beschrieben ist.
5. Wenn das SDK zu diesem Zeitpunkt noch keine Region gefunden hat, schlägt die Client-Erstellung mit einer Ausnahme fehl.

Bei der Entwicklung von AWS Anwendungen besteht ein häufiger Ansatz darin, die gemeinsame Konfigurationsdatei (in [der Reihenfolge des Abrufs von Anmeldeinformationen](#) beschrieben) zu verwenden, um die Region für die lokale Entwicklung festzulegen, und sich auf die standardmäßige Regionsanbieterkette zu verlassen, um die Region zu bestimmen, wenn die Anwendung auf der AWS Infrastruktur ausgeführt wird. Dies vereinfacht die Client-Erstellung stark und sorgt dafür, dass Ihre Anwendung portabel bleibt.

## Überprüfen der Serviceverfügbarkeit in einer Region

Um festzustellen, ob ein bestimmter in einer Region verfügbarer AWS-Service ist, verwenden Sie die `region` Methode `serviceMetadata` und auf dem Service-Client.

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

Weitere Informationen finden Sie in der Dokumentation zur [Region](#) Klasse für die , die AWS-Regionen Sie angeben können, und verwenden Sie das Endpunktpräfix des abzufragenden Services.

## Auswählen eines bestimmten Endpunkts

In bestimmten Situationen, z. B. um Vorschaufunktionen eines Services zu testen, bevor die Funktionen die allgemeine Verfügbarkeit erreichen, müssen Sie möglicherweise einen bestimmten Endpunkt in einer Region angeben. In diesen Situationen können Service-Clients durch Aufrufen der `-endpointOverride`-Methode konfiguriert werden.

Um beispielsweise einen Amazon EC2 Client für die Verwendung der Region Europa (Irland) mit einem bestimmten Endpunkt zu konfigurieren, verwenden Sie den folgenden Code.

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

Die aktuelle Liste der Regionen und ihrer entsprechenden Endpunkte für alle AWS Services finden Sie [unter Regionen und Endpunkte](#).

## Reduzieren Sie die Startzeit des SDK für AWS Lambda

Eines der Ziele von AWS SDK for Java 2.x besteht darin, die Startlatenz für AWS Lambda - Funktionen zu reduzieren. Das SDK enthält Änderungen, die die Startzeit reduzieren, die am Ende dieses Themas erörtert werden.

Zunächst konzentriert sich dieses Thema auf Änderungen, die Sie vornehmen können, um Kaltstartzeiten zu reduzieren. Dazu gehören Änderungen an Ihrer Codestruktur und an der Konfiguration von Service-Clients.

## Verwenden der SDKs `URLConnectionHttpClient`

Für synchrone Szenarien bietet das SDK for Java 2.x die `-URLConnectionHttpClient`-Klasse, die auf den HTTP-Clientklassen des JDK basiert. Da die auf bereits auf dem Klassenpfad basierenden Klassen `URLConnectionHttpClient` basiert, müssen keine zusätzlichen Abhängigkeiten geladen werden.

Informationen zum Hinzufügen der `URLConnectionHttpClient` zu Ihrem Lambda-Projekt und zum Konfigurieren seiner Verwendung finden Sie unter [Den URL-verbundungs-basierten HTTP-Client konfigurieren](#).

**Note**

Es gibt einige Funktionsbeschränkungen bei der `URLConnectionHttpClient` im Vergleich zur des SDK [ApacheHttpClient](#). `ApacheHttpClient` ist der standardmäßige asynchrone HTTP-Client im SDK. Beispielsweise unterstützt die `URLConnectionHttpClient` die HTTP-PATCH-Methode nicht.

Einige API AWS -Operationen erfordern PATCH-Anfragen. Diese Operationsnamen beginnen normalerweise mit `Update*`. Im Folgenden finden Sie mehrere Beispiele.

- [Mehrere Update\\* Operationen](#) in der AWS Security Hub -API und auch die [-BatchUpdateFindings](#) Operation
- Alle Amazon API Gateway-API-[Update\\* Operationen](#)
- [Mehrere Update\\* Operationen](#) in der Amazon WorkDocs API

Wenn Sie die verwenden könnten `URLConnectionHttpClient`, lesen Sie zunächst die API-Referenz für die AWS-Service, die Sie verwenden. Überprüfen Sie, ob die benötigten Operationen den PATCH-Vorgang verwenden.

## Verwenden der SDKs `AwsCrtAsyncHttpClient`

ist [AwsCrtAsyncHttpClient](#) das asynchrone Gegenstück zur Reduzierung der Lambda-Startzeit im SDK.

ist `AwsCrtAsyncHttpClient` ein asynchroner, nicht blockierender HTTP-Client. Es baut auf den Java-Bindungen der AWS Common Runtime auf, die in der Programmiersprache C geschrieben ist. Zu den Zielen bei der Entwicklung der AWS Common Runtime gehört die schnelle Leistung.

Der Abschnitt dieses Handbuchs zur [Konfiguration von HTTP-Clients](#) enthält Informationen zum Hinzufügen eines `AwsCrtAsyncHttpClient` zu Ihrem Lambda-Projekt und zum Konfigurieren seiner Verwendung.

## Entfernen ungenutzter HTTP-Client-Abhängigkeiten

Neben der expliziten Verwendung von `URLConnectionHttpClient` oder können `AwsCrtAsyncHttpClient` Sie andere HTTP-Clients entfernen, die das SDK standardmäßig einfügt. Die Lambda-Startzeit wird reduziert, wenn weniger Bibliotheken geladen werden müssen. Daher sollten Sie alle nicht verwendeten Artefakte entfernen, die die JVM laden muss.

Der folgende Ausschnitt einer Maven-pom.xml-Datei zeigt den Ausschluss des Apache-basierten HTTP-Clients und des Netty-basierten HTTP-Clients. (Diese Clients werden nicht benötigt, wenn Sie die `URLConnectionHttpClient` verwenden.) In diesem Beispiel werden die HTTP-Clients-Artefakte von der S3-Client-Abhängigkeit ausgeschlossen und das `url-connection-client` Artefakt hinzugefügt, das die `URLConnectionHttpClient` Klasse einfügt.

```
<project>
  <properties>
    <aws.java.sdk.version>2.17.290</aws.java.sdk.version>
  </properties>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>${aws.java.sdk.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>url-connection-client</artifactId>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>s3</artifactId>
      <exclusions>
        <exclusion>
          <groupId>software.amazon.awssdk</groupId>
          <artifactId>netty-nio-client</artifactId>
        </exclusion>
        <exclusion>
          <groupId>software.amazon.awssdk</groupId>
          <artifactId>apache-client</artifactId>
        </exclusion>
      </exclusions>
    </dependency>
  </dependencies>
</project>
```

Wenn Sie die verwenden `AwsCrtAsyncHttpClient`, ersetzen Sie die Abhängigkeit von `url-connection-client` eine Abhängigkeit von der `aws-crt-client`.

### Note

Fügen Sie das `<exclusions>` Element zu allen Service-Client-Abhängigkeiten in Ihrer `-pom.xml` Datei hinzu.

## Konfigurieren von Service-Clients für die Verknüpfung von Suchvorgängen

### Angeben einer Region

Wenn Sie einen Service-Client erstellen, rufen Sie die `-region` Methode im Service-Client-Builder auf. Dadurch wird der standardmäßige [Regionssuchprozess](#) des SDKs abgekürzt, der die AWS-Region Informationen an mehreren Stellen prüft.

Um den Lambda-Code unabhängig von der Region zu halten, verwenden Sie den folgenden Code innerhalb der `region` Methode. Dieser Code greift auf die vom Lambda-Container festgelegte `AWS_REGION` Umgebungsvariable zu.

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

### Verwenden der **EnvironmentVariableCredentialProvider**

Ähnlich wie das standardmäßige Suchverhalten für die Regionsinformationen sucht das SDK an mehreren Stellen nach Anmeldeinformationen. Indem Sie [EnvironmentVariableCredentialProvider](#) beim Erstellen eines Service-Clients die angeben, sparen Sie Zeit im Nachschlagevorgang des SDK.

### Note

Durch die Verwendung dieses Anmeldeinformationsanbieters kann der Code in - Lambda Funktionen verwendet werden, funktioniert aber möglicherweise nicht auf Amazon EC2 oder anderen Systemen.

Der folgende Codeausschnitt zeigt einen S3-Service-Client, der für die Verwendung in einer Lambda-Umgebung entsprechend konfiguriert ist.

```
S3Client client = S3Client.builder()

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

## Initialisieren des SDK-Clients außerhalb des Lambda-Funktionshandlers

Wir empfehlen, einen SDK-Client außerhalb der Lambda-Handler-Methode zu initialisieren. Wenn der Ausführungskontext wiederverwendet wird, kann die Initialisierung des Service-Clients auf diese Weise übersprungen werden. Durch die Wiederverwendung der Client-Instance und ihrer Verbindungen werden nachfolgende Aufrufe der Handler-Methode schneller ausgeführt.

Im folgenden Beispiel wird die `S3Client` Instance im Konstruktor mit einer statischen Factory-Methode initialisiert. Wenn der von der Lambda-Umgebung verwaltete Container wiederverwendet wird, wird die initialisierte `S3Client` Instance wiederverwendet.

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

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

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

## Minimieren der Abhängigkeitsinjektion

Frameworks für die Abhängigkeitsinjektion (DI) können zusätzliche Zeit in Anspruch nehmen, um den Einrichtungsprozess abzuschließen. Sie erfordern möglicherweise auch zusätzliche Abhängigkeiten, was das Laden dauert.

Wenn ein DI-Framework benötigt wird, empfehlen wir die Verwendung leichter DI-Frameworks wie [Dagger](#).

## Verwenden eines Maven-Arche-Typ-Targetings AWS Lambda

Das AWS Java SDK-Team hat eine [Maven Archetype](#)-Vorlage entwickelt, um ein Lambda-Projekt mit minimaler Startzeit zu booten. Sie können ein Maven-Projekt aus dem Archetyp erstellen und wissen, dass die Abhängigkeiten für die Lambda-Umgebung geeignet konfiguriert sind.

Weitere Informationen zum Archetyp und die Arbeit an einer Beispielbereitstellung finden Sie in diesem [Blogbeitrag](#).

## Lambda SnapStart für Java in Betracht ziehen

Wenn Ihre Laufzeitanforderungen kompatibel sind, AWS bietet [Lambda SnapStart für Java an](#). Lambda SnapStart ist eine infrastrukturbasierte Lösung, die die Startleistung für Java-Funktionen verbessert. Wenn Sie eine neue Version einer Funktion veröffentlichen, SnapStart initialisiert Lambda sie und erstellt einen unveränderlichen, verschlüsselten Snapshot des Speicher- und Festplattenstatus. SnapStart speichert den Snapshot zur Wiederverwendung zwischen.

## Änderungen der Version 2.x, die sich auf die Startzeit auswirken

Zusätzlich zu den Änderungen, die Sie an Ihrem Code vornehmen, enthält Version 2.x des SDK for Java drei primäre Änderungen, die die Startzeit reduzieren:

- Verwendung von [jackson-jr](#), einer Serialisierungsbibliothek, die die Initialisierungszeit verbessert
- Verwendung der [java.time](#)-Bibliotheken für Datums- und Uhrzeitobjekte, die Teil des JDK sind
- Verwenden von [Slf4j](#) für eine Protokollierungsfassade

## Weitere Ressourcen

Das AWS Lambda Entwicklerhandbuch enthält einen [Abschnitt zu bewährten Methoden](#) für die Entwicklung von Lambda-Funktionen, die nicht Java-spezifisch sind.

Ein Beispiel für die Erstellung einer cloudnativen Anwendung in Java, die verwendet AWS Lambda, finden Sie in diesem [Workshop-Inhalt](#). Die Leistungsoptimierung der Workshop-Diskussion und andere bewährte Methoden.

Sie können die Verwendung statischer Images in Betracht ziehen, die im Voraus kompiliert werden, um die Startlatenz zu reduzieren. Sie können beispielsweise das SDK for Java 2.x und Maven verwenden, um [ein natives GraalVM-Image zu erstellen](#).



# HTTP-Clients

Sie können den HTTP-Client ändern, der für Ihren Service-Client verwendet werden soll, sowie die Standardkonfiguration für HTTP-Clients mit dem `ändernAWS SDK for Java 2.x`. In diesem Abschnitt werden HTTP-Clients und Einstellungen für das SDK beschrieben.

## Im SDK for Java verfügbare HTTP-Clients

### Synchrone Clients

Synchrone HTTP-Clients im SDK for Java implementieren die [SdkHttpClient](#)-Schnittstelle. Ein synchroner Dienstclient, wie der `S3Client` oder der `DynamoDbClient`, erfordert die Verwendung eines synchronen HTTP-Clients. Der AWS SDK for Java bietet drei synchrone HTTP-Clients.

#### ApacheHttpClient (Standard)

[ApacheHttpClient](#) ist der Standard-HTTP-Client für synchrone Service-Clients. Hinweise zur Konfiguration von `finden Sie ApacheHttpClient` unter [Den Apache-basierten HTTP-Client konfigurieren](#).

#### AwsCrHttpClient

[AwsCrHttpClient](#) bietet hohen Durchsatz und blockierungsfreie I/O. Es basiert auf dem AWS Common Runtime (CRT) Http Client. Hinweise zur Konfiguration `AwsCrHttpClient` und Verwendung mit Service-Clients finden Sie unter [the section called "AWSCRT-basierte HTTP-Clients konfigurieren"](#).

#### URLConnectionHttpClient

Um die Anzahl der von Ihrer Anwendung verwendeten JAR-Dateien und Bibliotheken von Drittanbietern zu minimieren, können Sie die verwenden [URLConnectionHttpClient](#). Hinweise zur Konfiguration von `URLConnectionHttpClient` Sie unter [Den URL-verbindungs-basierten HTTP-Client konfigurieren](#).

### Asynchrone Clients

Asynchrone HTTP-Clients im SDK for Java implementieren die [SdkAsyncHttpClient](#)-Schnittstelle. Ein asynchroner Dienstclient, wie der `S3AsyncClient` oder der `DynamoDbAsyncClient`, erfordert die Verwendung eines asynchronen HTTP-Clients. Der AWS SDK for Java bietet zwei asynchrone HTTP-Clients.

## NettyNioAsyncHttpClient (Standard)

[NettyNioAsyncHttpClient](#) ist der Standard-HTTP-Client, der von asynchronen Clients verwendet wird. Hinweise zur Konfiguration von finden Sie [NettyNioAsyncHttpClient](#) unter [the section called "Konfigurieren des Netty-basierten HTTP-Clients"](#).

## AwsCrtAsyncHttpClient

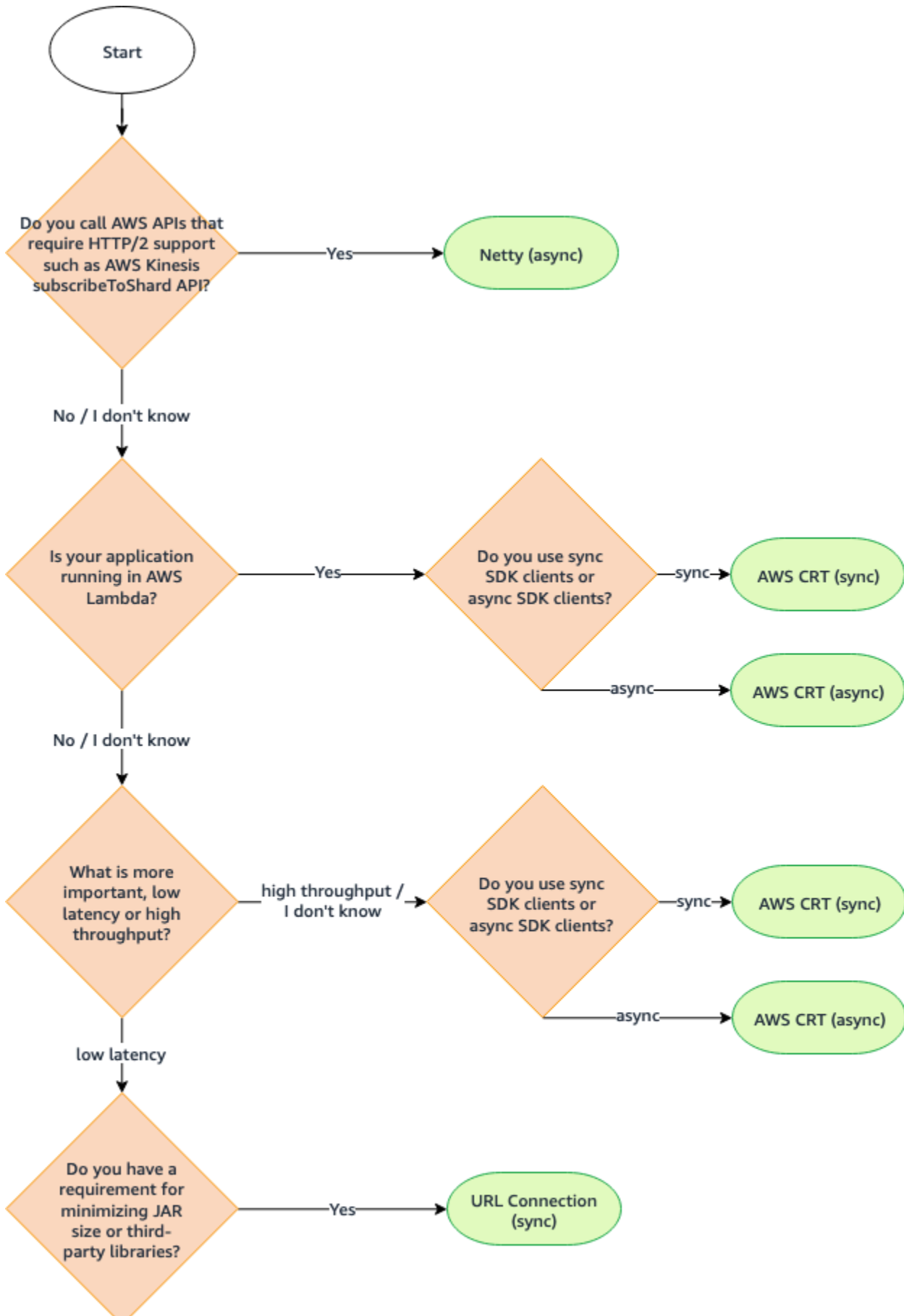
Der [AwsCrtAsyncHttpClient](#) basiert auf dem AWS Common Runtime (CRT) -HTTP-Client. Hinweise zur Konfiguration von finden Sie [AwsCrtAsyncHttpClient](#) unter [the section called "AWSCRT-basierte HTTP-Clients konfigurieren"](#).

## Empfehlungen für HTTP-Clients

Bei der Auswahl einer HTTP-Client-Implementierung spielen mehrere Faktoren eine Rolle. Verwenden Sie die folgenden Informationen als Entscheidungshilfe.

### Flussdiagramm für Empfehlungen

Das folgende Flussdiagramm enthält allgemeine Hinweise, anhand derer Sie bestimmen können, welcher HTTP-Client verwendet werden soll.



## Vergleich von HTTP-Clients

Die folgende Tabelle enthält detaillierte Informationen für jeden HTTP-Client.

HTTP-Client	Synchronisieren oder asynchron	Wann sollte dies verwendet werden?	Einschränkung/Nachteil
Apache-basierter HTTP-Client  (Standard-Synchronisierungs-HTTP-Client)	Synchronisierung	Verwenden Sie ihn, wenn Sie eine niedrige Latenz einem hohen Durchsatz vorziehen	Langsamere Startzeit im Vergleich zu anderen HTTP-Clients
Auf URL-Verbindung basierender HTTP-Client	Synchronisierung	Verwenden Sie ihn, wenn Sie unbedingt die Abhängigkeiten von Drittanbietern einschränken möchten	Unterstützt nicht die HTTP-PATCH-Methode, die für einige APIs wie Amazon APIGateway-Aktualisierungsvorgänge erforderlich ist
AWS <sup>CRT</sup> -basierter Sync-HTTP-Client 1	Synchronisierung	<ul style="list-style-type: none"> <li>• Verwenden Sie es, wenn Ihre Anwendung in läuft AWS Lambda</li> <li>• Verwenden Sie es, wenn Sie einen hohen Durchsatz einer niedrigen Latenz vorziehen</li> <li>• Verwenden Sie es, wenn Sie SDK-Clients synchronisieren möchten</li> </ul>	N/A
Netty-basierter HTTP-Client	Asynchron	• Verwenden Sie es, wenn Ihre Anwendung APIs aufruft, für die HTTP/2-Unterstützung erforderlich	Langsamere Startzeit im Vergleich zu

HTTP-Client	Synchronisieren oder asynchron	Wann sollte dies verwendet werden?	Einschränkung/ Nachteil
(standardmäßiger asynchroner HTTP-Client)		ist, z. B. die Kinesis-API <a href="#">SubscribeToShard</a>	anderen HTTP-Clients
AWS <sup>CRT</sup> -basierter asynchroner HTTP-Client <sup>1</sup>	Asynchron	<ul style="list-style-type: none"> <li>• Verwenden Sie es, wenn Ihre Anwendung in läuft AWS Lambda</li> <li>• Verwenden Sie es, wenn Sie einen hohen Durchsatz einer niedrigen Latenz vorziehen</li> <li>• Verwenden Sie es, wenn Sie asynchrone SDK-Clients bevorzugen</li> </ul>	<ul style="list-style-type: none"> <li>• Unterstützt keine Service-Clients, die HTTP/2-Unterstützung benötigen, wie und KinesisAsyncClient TranscribeStreamingAsyncClient</li> </ul>

<sup>1</sup> Aufgrund ihrer zusätzlichen Vorteile empfehlen wir, wenn möglich die AWS CRT-basierten HTTP-Clients zu verwenden.

## Standardeinstellungen für die intelligente Konfiguration

Die AWS SDK for Java 2.x (Version 2.17.102 oder höher) bietet eine Funktion für intelligente Standardeinstellungen für Konfigurationen. Diese Funktion optimiert zwei HTTP-Client-Eigenschaften zusammen mit anderen Eigenschaften, die den HTTP-Client nicht beeinflussen.

Die Standardeinstellungen der intelligenten Konfiguration legen sinnvolle Werte für die `tlsNegotiationTimeoutInMillis` Eigenschaften `connectTimeoutInMillis` und fest, die auf einem von Ihnen angegebenen Standardmoduswert basieren. Sie wählen den Standardmoduswert auf der Grundlage der Eigenschaften Ihrer Anwendung.

Weitere Informationen zu den Standardeinstellungen für intelligente Konfigurationen und zur Auswahl des Standardmoduswerts, der für Ihre Anwendungen am besten geeignet ist, finden Sie im Referenzhandbuch für [AWSSDKs und Tools](#).

Im Folgenden finden Sie vier Möglichkeiten, den Standardmodus für Ihre Anwendung festzulegen.

## Service client

Verwenden Sie den Service Client Builder, um den Standardmodus direkt auf dem Service Client zu konfigurieren. Im folgenden Beispiel wird der Standardmodus `auto` für den auf festgelegt.

### DynamoDbClient

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

## System property

Sie können die `aws.defaultsMode` Systemeigenschaft verwenden, um den Standardmodus anzugeben. Wenn Sie die Systemeigenschaft in Java festlegen, müssen Sie die Eigenschaft festlegen, bevor Sie einen Service-Client initialisieren.

Das folgende Beispiel zeigt Ihnen, wie Sie den Standardmodus so einstellen, dass er eine in `auto` Java festgelegte Systemeigenschaft verwendet.

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

Das folgende Beispiel zeigt, wie Sie den Standardmodus `auto` mithilfe einer `-D` Option des Befehls auf „Standardmodus“ festlegen. `java`

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

## Environment variable

Geben Sie einen Wert für die Umgebungsvariable `AWS_DEFAULTS_MODE`, um den Standardmodus für Ihre Anwendung auszuwählen.

Die folgenden Informationen zeigen den Befehl, der ausgeführt werden muss, um den Wert für den Standardmodus auf die `auto` Verwendung einer Umgebungsvariablen festzulegen.

Betriebssystem	Befehl zum Setzen von Umgebungsvariablen
Linux, macOS oder Unix	<code>export AWS_DEFAULTS_MODE=auto</code>

Betriebssystem	Befehl zum Setzen von Umgebungsvariablen
Windows	<code>set AWS_DEFAULTS_MODE=auto</code>

## AWS config file

Sie können der gemeinsam genutzten AWS config Datei eine `defaults_mode` Konfigurationseigenschaft hinzufügen, wie das folgende Beispiel zeigt.

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

Wenn Sie den Standardmodus global mit der Systemeigenschaft, der Umgebungsvariablen oder der AWS Konfigurationsdatei festlegen, können Sie die Einstellungen beim Erstellen eines HTTP-Clients überschreiben.

Wenn Sie mit `httpClientBuilder()` dieser Methode einen HTTP-Client erstellen, gelten die Einstellungen nur für die Instanz, die Sie gerade erstellen. Ein Beispiel dafür wird [hier](#) gezeigt. Der Netty-basierte HTTP-Client in diesem Beispiel überschreibt alle global für und festgelegten Standardmoduswerte. `connectTimeoutInMillis` `tlsNegotiationTimeoutInMillis`

## Proxy-Unterstützung

Sie können HTTP-Proxys konfigurieren, indem Sie Code verwenden, Java-Systemeigenschaften festlegen oder beide Ansätze kombinieren. Das SDK unterstützt derzeit keine Umgebungsvariablen zur Konfiguration von Proxys.

Sie konfigurieren Proxys im Code mit einem kundenspezifischen `ProxyConfiguration` Builder, wenn Sie den Service-Client erstellen. Der Abschnitt für jeden HTTP-Client in diesem Thema enthält ein Beispiel für eine Proxykonfiguration. Dieses [Beispiel bezieht sich auf den Apache HTTP-Client](#).

### HTTP-Client-Unterstützung von Java-Systemeigenschaften für HTTP-Proxys

Systemeigenschaft	Beschreibung	Unterstützung für HTTP-Clients
<code>http.ProxyHost</code>	Hostname des HTTP-Proxyservers	Alle

Systemeigenschaft	Beschreibung	Unterstützung für HTTP-Clients
http.ProxyPort	Portnummer des HTTP-Proxyservers	Alle
http.ProxyUser	Benutzername für die HTTP-Proxyauthentifizierung	Alle
http.ProxyPassword	Passwort für die HTTP-Proxyauthentifizierung	Alle
http.nonProxyHosts	Liste der Hosts, die unter Umgehung des Proxys direkt erreicht werden sollen. <a href="#">Gilt auch, wenn HTTPS verwendet wird.</a>	Alle
https.proxyHost	Hostname des HTTPS-Proxyservers	Netty, CRT
HTTPS.Proxy-Port	Portnummer des HTTPS-Proxyservers	Netty, CRT
https.proxyBenutzer	Benutzername für die HTTPS-Proxyauthentifizierung	Netty, CRT
https.proxyPasswort	Passwort für die HTTPS-Proxyauthentifizierung	Netty, CRT

Die in den Tabellen verwendeten Begriffe bedeuten:

- **Alle:** Alle vom SDK angebotenen HTTP-Clients  
—`URLConnectionHttpClient`, `ApacheHttpClient`, `NettyNioAsyncHttpClient`, `AwsCrtAsyncHttpClient`
- **Netty:** der Netty-basierte HTTP-Client () `NettyNioAsyncHttpClient`
- **CRT:** die AWS CRT-basierten HTTP-Clients, (und) `AwsCrtHttpClient` `AwsCrtAsyncHttpClient`



Sie können eine Mischung aus HTTP-Clientkonfiguration und Systemeigenschaften verwenden. Der `ProxyConfiguration` Builder jedes HTTP-Clients bietet eine `useSystemPropertyValues` Einstellung. Standardmäßig ist die Einstellung `true`. Wenn die Einstellung aktiviert ist `true`, verwendet das SDK automatisch Systemeigenschaftswerte für Optionen, die nicht mithilfe des `ProxyConfiguration` Builders bereitgestellt werden.

Das folgende Beispiel zeigt die Konfiguration, die durch eine Systemeigenschaft und durch Code bereitgestellt wird.

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

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

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

### Note

Anstatt die `endpoint` Eigenschaft im Code festzulegen, wie im vorherigen Codeausschnitt gezeigt, können Sie die folgenden Systemeigenschaften verwenden.

```
-Dhttp.proxyHost=localhost -Dhttp.proxyPort=1234
```

## Den Apache-basierten HTTP-Client konfigurieren

Synchrone Service-Clients AWS SDK for Java 2.x verwenden standardmäßig einen Apache-basierten HTTP-Client. [ApacheHttpClient](#) Die SDKs basieren auf dem `ApacheHttpClient` Klasse.

Das SDK bietet auch das [URLConnectionHttpClient](#), das schneller geladen wird, aber weniger Funktionen hat. Hinweise zur Konfiguration von `URLConnectionHttpClient` Sie unter [the section called "Den URL-verbundungs-basierten HTTP-Client konfigurieren"](#).

Alle verfügbaren Konfigurationsoptionen für finden Sie unter [ApacheHttpClient.Builder und ProxyConfiguration.Builder](#). `ApacheHttpClient`

### Greifen Sie auf `ApacheHttpClient`

In den meisten Situationen verwenden Sie die `ApacheHttpClient` ohne explizite Konfiguration. Sie deklarieren Ihre Service-Clients und das SDK konfiguriert sie `ApacheHttpClient` mit Standardwerten für Sie.

Wenn Sie den explizit konfigurieren `ApacheHttpClient` oder ihn mit mehreren Service-Clients verwenden möchten, müssen Sie ihn für die Konfiguration verfügbar machen.

Keine Konfiguration erforderlich

Wenn Sie in Maven eine Abhängigkeit von einem Service-Client deklarieren, fügt das SDK eine Laufzeitabhängigkeit von dem `apache-client` Artefakt hinzu. Dadurch steht die `ApacheHttpClient` Klasse Ihrem Code zur Laufzeit zur Verfügung, aber nicht zur Kompilierzeit. Wenn Sie den Apache-basierten HTTP-Client nicht konfigurieren, müssen Sie dafür keine Abhängigkeit angeben.

Im folgenden XML-Snippet einer `pom.xml` Maven-Datei wird durch die mit deklarierte Abhängigkeit `<artifactId>s3</artifactId>` automatisch der Apache-basierte HTTP-Client eingebunden. Sie müssen keine spezielle Abhängigkeit dafür deklarieren.

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

```
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <!-- The s3 dependency automatically adds a runtime dependency on the
    ApacheHttpClient-->
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>s3</artifactId>
    </dependency>
</dependencies>
```

Mit diesen Abhängigkeiten können Sie keine expliziten Änderungen an der HTTP-Konfiguration vornehmen, da sich die `ApacheHttpClient` Bibliothek nur im Laufzeit-Klassenpfad befindet.

### Konfiguration erforderlich

Um das zu konfigurieren `ApacheHttpClient`, müssen Sie bei der Kompilierung eine Abhängigkeit von der `apache-client` Bibliothek hinzufügen.

Sehen Sie sich das folgende Beispiel für eine `pom.xml` Maven-Datei an, um die `ApacheHttpClient` zu konfigurieren.

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

```
<artifactId>apache-client</artifactId>
</dependency>
</dependencies>
```

## Verwenden und konfigurieren Sie den **ApacheHttpClient**

Sie können eine Instanz von `ApacheHttpClient` konfigurieren und gleichzeitig einen Service Client erstellen, oder Sie können eine einzelne Instanz so konfigurieren, dass sie von mehreren Service Clients gemeinsam genutzt wird.

Bei beiden Ansätzen verwenden Sie den, [ApacheHttpClient.Builder](#) um die Eigenschaften für den Apache-basierten HTTP-Client zu konfigurieren.

Bewährtes Verfahren: Weisen Sie einem Service-Client eine **ApacheHttpClient** Instanz zu

Wenn Sie eine Instanz von konfigurieren müssen, empfehlen wir Ihnen `ApacheHttpClient`, die dedizierte `ApacheHttpClient` Instanz zu erstellen. Sie können dies tun, indem Sie die `httpClientBuilder` Methode des Builders des Service-Clients verwenden. Auf diese Weise wird der Lebenszyklus des HTTP-Clients vom SDK verwaltet, wodurch potenzielle Speicherlecks vermieden werden, wenn die `ApacheHttpClient` Instanz nicht geschlossen wird, wenn sie nicht mehr benötigt wird.

Im folgenden Beispiel wird eine `S3Client` eingebettete Instanz von `ApacheHttpClient` with `maxConnections` and `connectionTimeout` values erstellt und konfiguriert. Die HTTP-Instanz wird mit der `httpClientBuilder` Methode von `S3Client.Builder` erstellt.

### Importe

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

### Code

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

// Perform work with the s3Client.
```

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

## Alternativer Ansatz: Eine **ApacheHttpClient** Instanz teilen

Um den Ressourcen- und Speicherverbrauch für Ihre Anwendung zu senken, können Sie eine konfigurieren `ApacheHttpClient` und sie von mehreren Service-Clients gemeinsam nutzen. Der HTTP-Verbindungspool wird gemeinsam genutzt, wodurch die Ressourcennutzung gesenkt wird.

### Note

Wenn eine `ApacheHttpClient` Instanz gemeinsam genutzt wird, müssen Sie sie schließen, wenn sie bereit ist, gelöscht zu werden. Das SDK schließt die Instanz nicht, wenn der Service-Client geschlossen wird.

Im folgenden Beispiel wird ein Apache-basierter HTTP-Client konfiguriert, der von zwei Service-Clients verwendet wird. Die konfigurierte `ApacheHttpClient` Instanz wird an die `httpClient` Methode jedes Builders übergeben. Wenn die Service-Clients und der HTTP-Client nicht mehr benötigt werden, werden sie durch den Code explizit geschlossen. Der Code schließt den HTTP-Client zuletzt.

## Importe

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

## Code

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

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

DynamoDbClient dynamoDbClient =
    DynamoDbClient.builder()
```

```
        .httpClient(apacheHttpClient).build();

// Perform work with the s3Client and dynamoDbClient.

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

## Beispiel für eine Proxykonfiguration

Der folgende Codeausschnitt verwendet den [Proxykonfigurationsgenerator für den Apache HTTP-Client](#).

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

Die entsprechenden Java-Systemeigenschaften für die Proxykonfiguration werden im folgenden Befehlszeilenausschnitt angezeigt.

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

### Note

Der Apache HTTP-Client unterstützt derzeit keine HTTPS-Proxy-Systemeigenschaften.

## Den URL-verbindungsbasierten HTTP-Client konfigurieren

Der AWS SDK for Java 2.x bietet im Vergleich zum Standard einen leichteren [URLConnectionHttpClient](#) HTTP-Client. `ApacheHttpClient` Der `URLConnectionHttpClient` basiert auf dem von Java. [URLConnection](#)

Der `URLConnectionHttpClient` wird schneller geladen als der Apache-basierte HTTP-Client, hat aber weniger Funktionen. Da er schneller geladen wird, ist er eine [gute Lösung](#) für Java-Funktionen.

## AWS Lambda

Das `URLConnectionHttpClient` hat mehrere [konfigurierbare Optionen](#), auf die Sie zugreifen können.

### Note

Der `URLConnectionHttpClient` unterstützt die HTTP-PATCH-Methode nicht. Für eine Handvoll AWS API-Operationen sind PATCH-Anfragen erforderlich. Diese Operationsnamen beginnen normalerweise mit `Update*`. Im Folgenden finden Sie einige Beispiele.

- [Verschiedene Update\\* Operationen](#) in der AWS Security Hub API und auch die [BatchUpdateFindingsOperation](#)
- Alle [Update\\*API-Operationen](#) von Amazon API Gateway
- [Verschiedene Update\\* Operationen](#) in der WorkDocs Amazon-API

Wenn Sie die verwenden könnten `URLConnectionHttpClient`, lesen Sie zunächst in der API-Referenz nach AWS-Service, die Sie verwenden. Prüfen Sie, ob die Operationen, die Sie benötigen, den PATCH-Vorgang verwenden.

## Greifen Sie auf `URLConnectionHttpClient`

Um das zu konfigurieren und zu verwenden `URLConnectionHttpClient`, deklarieren Sie eine Abhängigkeit vom `url-connection-client` Maven-Artefakt in Ihrer `pom.xml` Datei.

Im Gegensatz zu `URLConnectionHttpClient` wird das nicht automatisch zu Ihrem Projekt hinzugefügt, daher muss es von `use` ausdrücklich deklariert werden. `ApacheHttpClient`

Das folgende Beispiel einer `pom.xml` Datei zeigt die Abhängigkeiten, die für die Verwendung und Konfiguration des HTTP-Clients erforderlich sind.

```
<dependencyManagement>
  <dependencies>
    <dependency>
```

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

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

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

## Verwenden und konfigurieren Sie **URLConnectionHttpClient**

Sie können eine Instanz von `URLConnectionHttpClient` konfigurieren und gleichzeitig einen Service Client erstellen, oder Sie können eine einzelne Instanz so konfigurieren, dass sie von mehreren Service Clients gemeinsam genutzt wird.

Bei beiden Ansätzen verwenden Sie den [URLConnectionHttpClient.Builder](#), um die Eigenschaften für den URLConnection-basierten HTTP-Client zu konfigurieren.

Bewährtes Verfahren: Dedizieren Sie eine **URLConnectionHttpClient** Instanz einem Service-Client

Wenn Sie eine Instanz von konfigurieren müssen, empfehlen wir Ihnen `URLConnectionHttpClient`, die dedizierte `URLConnectionHttpClient` Instanz zu erstellen. Sie können dies tun, indem Sie die `httpClientBuilder` Methode des Builders des Service-Clients verwenden. Auf diese Weise wird der Lebenszyklus des HTTP-Clients vom SDK verwaltet, wodurch potenzielle Speicherlecks vermieden werden, wenn die `URLConnectionHttpClient` Instanz nicht geschlossen wird, wenn sie nicht mehr benötigt wird.

Im folgenden Beispiel wird eine `S3Client` eingebettete Instanz von `URLConnectionHttpClient` with `socketTimeout` and `proxyConfiguration` values erstellt und konfiguriert. Die `proxyConfiguration` Methode verwendet einen Java-Lambda-Ausdruck vom Typ `Consumer<ProxyConfiguration.Builder>`



## Importe

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

## Code

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

// Perform work with the s3Client.

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

### Alternativer Ansatz: Eine **UrlConnectionHttpClient** Instanz teilen

Um den Ressourcen- und Speicherverbrauch für Ihre Anwendung zu senken, können Sie eine konfigurieren `UrlConnectionHttpClient` und sie von mehreren Service-Clients gemeinsam nutzen. Der HTTP-Verbindungspool wird gemeinsam genutzt, was die Ressourcennutzung senkt.

#### Note

Wenn eine `UrlConnectionHttpClient` Instanz gemeinsam genutzt wird, müssen Sie sie schließen, wenn sie bereit ist, gelöscht zu werden. Das SDK schließt die Instanz nicht, wenn der Service-Client geschlossen wird.

Im folgenden Beispiel wird ein auf einer URL-Verbindung basierender HTTP-Client konfiguriert, der von zwei Dienstclients verwendet wird. Die konfigurierte `UrlConnectionHttpClient` Instanz wird an die `httpClient` Methode jedes Builders übergeben. Wenn die Service-Clients und der HTTP-Client nicht mehr benötigt werden, werden sie durch den Code explizit geschlossen. Der Code schließt den HTTP-Client zuletzt.

## Importe

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

## Code

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

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

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

// Perform work with the s3Client and dynamoDbClient.

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

Verwenden Sie **UrlConnectionHttpClient** und **ApacheHttpClient** zusammen

Wenn Sie das `UrlConnectionHttpClient` in Ihrer Anwendung verwenden, müssen Sie jedem Service-Client entweder eine `UrlConnectionHttpClient` Instanz oder eine `ApacheHttpClient`

Instanz mithilfe der `httpClientBuilder` Methode des Service Client Builders zur Verfügung stellen.

Eine Ausnahme tritt auf, wenn Ihr Programm mehrere Service-Clients verwendet und beide der folgenden Bedingungen zutreffen:

- Ein Dienstclient ist für die Verwendung einer `URLConnectionHttpClient` Instanz konfiguriert
- Ein anderer Dienstclient verwendet den Standard, `ApacheHttpClient` ohne ihn explizit mit den `httpClientBuilder()` Methoden `httpClient()` oder `zu erstellen`

Die Ausnahme besagt, dass im Klassenpfad mehrere HTTP-Implementierungen gefunden wurden.

Der folgende Beispielausschnitt führt zu einer Ausnahme.

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

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

// Perform work with the s3Client and dynamoDbClient.

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

Vermeiden Sie die Ausnahme, indem Sie das explizit `S3Client` mit einem konfigurieren.

`ApacheHttpClient`

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

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

```
// Perform work with the s3Client and dynamoDbClient.  
  
dynamoDbClient.close();  
s3Client.close();
```

### Note

Um das explizit zu erstellen `ApacheHttpClient`, müssen Sie Ihrer Maven-Projektdatei [eine Abhängigkeit vom `apache-client` Artefakt hinzufügen](#).

## Beispiel für eine Proxy-Konfiguration

Der folgende Codeausschnitt verwendet den [Proxykonfigurationsgenerator für den URL-Verbindungs-HTTP-Client](#).

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

Die entsprechenden Java-Systemeigenschaften für die Proxykonfiguration werden im folgenden Befehlszeilenausschnitt angezeigt.

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

### Note

Der HTTP-Client für die URL-Verbindung unterstützt derzeit keine HTTPS-Proxy-Systemeigenschaften.

## Konfigurieren des Netty-basierten HTTP-Clients

Der Standard-HTTP-Client für asynchrone Operationen in der AWS SDK for Java 2.x ist der netzbasierte [NettyNioAsyncHttpClient](#). Der Netty-basierte Client basiert auf dem asynchronen ereignisgesteuerten Netzwerk-Framework des [Netty-Projekts](#).

Als alternativer HTTP-Client können Sie den neuen [AWS CRT-basierten HTTP-Client](#) verwenden. In diesem Thema erfahren Sie, wie Sie die konfigurieren [NettyNioAsyncHttpClient](#).

### Zugriff auf die **NettyNioAsyncHttpClient**

In den meisten Situationen verwenden Sie die [NettyNioAsyncHttpClient](#) ohne explizite Konfiguration in asynchronen Programmen. Sie deklarieren Ihre asynchronen Service-Clients und das SDK konfiguriert die [NettyNioAsyncHttpClient](#) mit Standardwerten für Sie.

Wenn Sie die explizit konfigurieren [NettyNioAsyncHttpClient](#) oder mit mehreren Service-Clients verwenden möchten, müssen Sie sie für die Konfiguration verfügbar machen.

#### Keine Konfiguration erforderlich

Wenn Sie eine Abhängigkeit von einem Service-Client in Maven deklarieren, fügt das SDK eine Laufzeitabhängigkeit vom `netty-nio-client` Artefakt hinzu. Dadurch wird die [NettyNioAsyncHttpClient](#) Klasse zur Laufzeit für Ihren Code verfügbar, jedoch nicht zur Kompilierungszeit. Wenn Sie den Netty-basierten HTTP-Client nicht konfigurieren, müssen Sie keine Abhängigkeit dafür angeben.

Im folgenden XML-Ausschnitt einer Maven-`pom.xml`-Datei führt die mit deklarierte Abhängigkeit `<artifactId>dynamodb-enhanced</artifactId>` transitiv zum Netty-basierten HTTP-Client. Sie müssen keine Abhängigkeit speziell dafür deklarieren.

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

```
<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>dynamodb-enhanced</artifactId>
  </dependency>
</dependencies>
```

Mit diesen Abhängigkeiten können Sie keine HTTP-Konfigurationsänderungen vornehmen, da sich die `NettyNioAsyncHttpClient` Bibliothek nur auf dem Klassenpfad der Laufzeit befindet.

### Konfiguration erforderlich

Um die zu konfigurieren `NettyNioAsyncHttpClient`, müssen Sie zum Zeitpunkt der Kompilierung eine Abhängigkeit vom `netty-nio-client` Artefakt hinzufügen.

Weitere Informationen finden Sie im folgenden Beispiel einer `Maven-pom.xml` Datei zum Konfigurieren der `NettyNioAsyncHttpClient`.

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

## Verwenden und Konfigurieren der **NettyNioAsyncHttpClient**

Sie können eine Instance von `NettyNioAsyncHttpClient` zusammen mit der Erstellung eines Service-Clients konfigurieren oder eine einzelne Instance für die gemeinsame Nutzung über mehrere Service-Clients hinweg konfigurieren.

Bei beiden Ansätzen verwenden Sie den [NettyNioAsyncHttpClient.Builder](#), um die Eigenschaften für die Netty-basierte HTTP-Client-Instance zu konfigurieren.

**Bewährte Methode:** Eine **NettyNioAsyncHttpClient** Instance einem Service-Client zuweisen

Wenn Sie eine Instance des konfigurieren müssen `NettyNioAsyncHttpClient`, empfehlen wir Ihnen, eine dedizierte `NettyNioAsyncHttpClient` Instance zu erstellen. Sie können dies tun, indem Sie die `httpClientBuilder` Methode des Builders des Service-Clients verwenden. Auf diese Weise wird der Lebenszyklus des HTTP-Clients vom SDK verwaltet, wodurch potenzielle Speicherlecks vermieden werden, wenn die `NettyNioAsyncHttpClient` Instance nicht geschlossen wird, wenn sie nicht mehr benötigt wird.

Im folgenden Beispiel wird eine `DynamoDbAsyncClient` Instance erstellt, die von einer `DynamoDbEnhancedAsyncClient` Instance verwendet wird. Die `DynamoDbAsyncClient` Instance enthält die `NettyNioAsyncHttpClient` Instance mit den `maxConcurrency` Werten `connectionTimeout` und `httpClientBuilder` Methode erstellt `DynamoDbAsyncClient.Builder`.

### Importe

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

### Code

```
// DynamoDbAsyncClient is the lower-level client used by the enhanced client.
DynamoDbAsyncClient dynamoDbAsyncClient =
    DynamoDbAsyncClient
        .builder()
```

```
.httpClientBuilder(NettyNioAsyncHttpClient.builder()
    .connectionTimeout(Duration.ofMillis(5_000))
    .maxConcurrency(100)
    .tlsNegotiationTimeout(Duration.ofMillis(3_500)))
    .defaultsMode(DefaultsMode.IN_REGION)
    .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
    .build());

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

// Perform work with the dynamoDbAsyncClient and enhancedClient.

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

### Alternativer Ansatz: Freigeben einer **NettyNioAsyncHttpClient** Instance

Um die Ressourcen- und Speicherauslastung für Ihre Anwendung niedrig zu halten, können Sie eine konfigurieren `NettyNioAsyncHttpClient` und sie für mehrere Service-Clients freigeben. Der HTTP-Verbindungspool wird freigegeben, wodurch die Ressourcennutzung verringert wird.

#### Note

Wenn eine `NettyNioAsyncHttpClient` Instance freigegeben wird, müssen Sie sie schließen, wenn sie bereit ist, gehindert zu werden. Das SDK schließt die Instance nicht, wenn der Service-Client geschlossen wird.

Im folgenden Beispiel wird ein Netty-basierter HTTP-Client konfiguriert, der von zwei Service-Clients verwendet wird. Die `NettyNioAsyncHttpClient` konfigurierte Instance wird an die `httpClient` Methode jedes Builders übergeben. Wenn die Service-Clients und der HTTP-Client nicht mehr benötigt werden, schließt der Code sie explizit. Der Code schließt den HTTP-Client zuletzt.

#### Importe

```
import software.amazon.awssdk.http.SdkHttpClient;
```



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

## Code

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

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

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

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

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

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

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

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

## Beispiel für eine Proxy-Konfiguration

Der folgende Codeausschnitt verwendet den [Proxy-Konfigurations-Builder für den Netty-HTTP-Client](#)

```
SdkAsyncHttpClient nettyHttpClient = NettyNioAsyncHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .scheme("https")
        .host("myproxy")
        .port(1234)
        .username("username")
        .password("password")
        .build())
    .build();
```

Die entsprechenden Java-Systemeigenschaften für die Proxy-Konfiguration werden im folgenden Befehlszeilenausschnitt angezeigt.

```
$ java -Dhttps.proxyHost=myproxy -Dhttps.proxyPort=1234 -Dhttps.proxyUser=username \
-Dhttps.proxyPassword=password -cp ... App
```

### Important

Um eine der HTTPS-Proxy-Systemeigenschaften verwenden zu können, muss die `scheme` Eigenschaft im Code auf `gesetzt` sein `https`. Wenn die `Schemaeigenschaft` nicht im Code festgelegt ist, ist das Schema standardmäßig `HTTP` und das SDK sucht nur nach `http.*` Systemeigenschaften.

## AWSCRT-basierte HTTP-Clients konfigurieren

Zu den AWS CRT-basierten HTTP-Clients gehören synchrone und asynchrone [AwsCrtHttpClient](#) Clients. [AwsCrtAsyncHttpClient](#) Die AWS CRT-basierten HTTP-Clients bieten die folgenden Vorteile für HTTP-Clients:

- Schnellere SDK-Startzeit
- Geringerer Speicherbedarf
- Reduzierte Latenzzeit
- Verwaltung der Verbindungsintegrität
- DNS-Lastenausgleich

### AWSCRT-basierte Komponenten im SDK

Die in diesem Thema beschriebenen AWS CRT-basierten HTTP-Clients und der AWS CRT-basierte S3-Client sind unterschiedliche Komponenten im SDK.

Die synchronen und asynchronen AWSCRT-basierten HTTP-Clients sind SDK-HTTP-Client-Schnittstellen für Implementierungen und werden für die allgemeine HTTP-Kommunikation verwendet. Sie sind Alternativen zu den anderen synchronen oder asynchronen HTTP-Clients im SDK mit zusätzlichen Vorteilen.

Der [AWSCRT-basierte S3-Client](#) ist eine Implementierung der [AsyncClientS3-Schnittstelle](#) und wird für die Arbeit mit dem Amazon S3 S3-Service verwendet. Es ist eine Alternative zur Java-basierten Implementierung der `S3AsyncClient` Schnittstelle und bietet mehrere Vorteile.

Obwohl beide Komponenten Bibliotheken aus der [AWSCommon Runtime](#) verwenden, verwenden die AWS CRT-basierten HTTP-Clients die [aws-c-s3-Bibliothek](#) nicht und unterstützen auch nicht die API-Funktionen für [mehnteilige Uploads von S3](#). Der AWS CRT-basierte S3-Client wurde dagegen speziell für die Unterstützung der API-Funktionen für mehnteilige S3-Uploads entwickelt.

## Greifen Sie auf die CRT-basierten HTTP-Clients zu AWS

Bevor Sie die AWS CRT-basierten HTTP-Clients verwenden können, fügen Sie das `aws-crt-client` Artefakt mit einer Mindestversion von 2.22.0 zu den Abhängigkeiten Ihres Projekts hinzu.

Der folgende Maven `pom.xml` zeigt die AWS CRT-basierten HTTP-Clients, die mithilfe des Stücklistenmechanismus (BOM) deklariert wurden.

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

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>aws-crt-client</artifactId>
</dependency>
</dependencies>
</project>
```

[Die neueste Version finden Sie im zentralen Maven-Repository.](#)

## Verwenden und konfigurieren Sie einen AWS CRT-basierten HTTP-Client

Sie können einen AWS CRT-basierten HTTP-Client zusammen mit der Erstellung eines Service-Clients konfigurieren, oder Sie können eine einzelne Instanz so konfigurieren, dass sie von mehreren Service-Clients gemeinsam genutzt wird.

Bei beiden Ansätzen verwenden Sie einen Builder, um [die Eigenschaften für die AWS CRT-basierte HTTP-Clientinstanz zu konfigurieren](#).

**Bewährtes Verfahren:** Dedizieren Sie eine Instanz einem Service-Client

Wenn Sie eine Instanz eines AWS CRT-basierten HTTP-Clients konfigurieren müssen, empfehlen wir, dass Sie die Instanz dedizieren, indem Sie sie zusammen mit dem Service-Client erstellen. Sie können dies tun, indem Sie die `httpClientBuilder` Methode des Builders des Service-Clients verwenden. Auf diese Weise wird der Lebenszyklus des HTTP-Clients vom SDK verwaltet, wodurch potenzielle Speicherlecks vermieden werden, wenn die AWS CRT-basierte HTTP-Client-Instanz nicht geschlossen wird, wenn sie nicht mehr benötigt wird.

Im folgenden Beispiel wird ein S3-Serviceclient erstellt und ein AWS CRT-basierter HTTP-Client mit Werten und konfiguriert. `connectionTimeout` `maxConcurrency`

Synchronous client

Importe

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

Code

```
// Singleton: Use s3Client for all requests.
```

```
S3Client s3Client = S3Client.builder()
    .httpClientBuilder(AwsCrtHttpClient
        .builder()
        .connectionTimeout(Duration.ofSeconds(3))
        .maxConcurrency(100))
    .build();

// Perform work with the s3Client.

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

## Asynchronous client

### Importe

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

### Code

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

// Perform work with the s3AsyncClient.

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

## Alternativer Ansatz: Eine Instanz teilen

Um die Ressourcen- und Speicherauslastung für Ihre Anwendung gering zu halten, können Sie einen AWS CRT-basierten HTTP-Client konfigurieren und ihn von mehreren Service-Clients gemeinsam nutzen. Der HTTP-Verbindungspool wird gemeinsam genutzt, was die Ressourcennutzung senkt.

**Note**

Wenn eine AWS CRT-basierte HTTP-Client-Instanz gemeinsam genutzt wird, müssen Sie sie schließen, wenn sie bereit ist, gelöscht zu werden. Das SDK schließt die Instanz nicht, wenn der Service-Client geschlossen wird.

Im folgenden Beispiel wird eine AWS CRT-basierte HTTP-Clientinstanz mit `connectionTimeout` Werten und konfiguriert. `maxConcurrency` Die konfigurierte Instanz wird an die `httpClient` Methode des Builders jedes Service-Clients übergeben. Wenn die Service-Clients und der HTTP-Client nicht mehr benötigt werden, werden sie explizit geschlossen. Der HTTP-Client wird zuletzt geschlossen.

**Synchronous client****Importe**

```
import software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.awscore.defaultsmode.DefaultsMode;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.crt.AwsCrtHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.s3.S3Client;
import java.time.Duration;
```

**Code**

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

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

```
.build();

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

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

## Asynchronous client

### Importe

```
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.awscore.defaultsmode.DefaultsMode;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.crt.AwsCrtAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import java.time.Duration;
```

### Code

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

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

```

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

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

```

## Stellen Sie einen AWS CRT-basierten HTTP-Client als Standard ein

Sie können Ihre Maven-Build-Datei so einrichten, dass das SDK einen AWS CRT-basierten HTTP-Client als Standard-HTTP-Client für Service-Clients verwendet.

Dazu fügen Sie jedem Service-Client-Artefakt ein `exclusions` Element mit den standardmäßigen HTTP-Client-Abhängigkeiten hinzu.

Im folgenden `pom.xml` Beispiel verwendet das SDK einen AWS CRT-basierten HTTP-Client für S3-Dienste. Wenn der Service-Client in Ihrem Code ein `istS3AsyncClient` ist, verwendet das SDK `AwsCrtAsyncHttpClient`. Wenn der Service-Client ein `S3Client` ist, verwendet das SDK `AwsCrtHttpClient`. Bei diesem Setup sind der standardmäßige asynchrone HTTP-Client auf Netty-Basis und der standardmäßige asynchrone HTTP-Client auf Apache-Basis nicht verfügbar.

```

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

```



```
        <exclusion>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>apache-client</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>aws-crt-client</artifactId>
</dependency>
</dependencies>
</project>
```

*Den neuesten VERSION-Wert finden Sie im zentralen Maven-Repository.*

### Note

Wenn mehrere Service-Clients in einer pom.xml Datei deklariert sind, benötigen alle das exclusions XML-Element.

Verwenden Sie eine Java-Systemeigenschaft

Um die AWS CRT-basierten HTTP-Clients als Standard-HTTP für Ihre Anwendung zu verwenden, können Sie die Java-Systemeigenschaft `software.amazon.awssdk.http.async.service.impl` auf den Wert von `software.amazon.awssdk.http.crt.AwsCrtSdkHttpClient` setzen.

Um den Wert beim Start der Anwendung festzulegen, führen Sie einen Befehl aus, der dem folgenden ähnelt.

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

Verwenden Sie den folgenden Codeausschnitt, um die Systemeigenschaft in Ihrem Anwendungscode festzulegen.

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

**Note**

Sie müssen eine Abhängigkeit von dem `aws-crt-client` Artefakt in Ihrer `pom1.xml` Datei hinzufügen, wenn Sie eine Systemeigenschaft verwenden, um die Verwendung der AWS CRT-basierten HTTP-Clients zu konfigurieren.

## Erweiterte Konfiguration von CRT-basierten HTTP-Clients AWS

Sie können verschiedene Konfigurationseinstellungen der AWS CRT-basierten HTTP-Clients verwenden, einschließlich der Konfiguration des Verbindungsstatus und der maximalen Leerlaufzeit. Sie können die [verfügbaren Konfigurationsoptionen](#) für überprüfen. `AwsCrtAsyncHttpClient` Sie können dieselben Optionen für den konfigurieren `AwsCrtHttpClient`.

### Konfiguration des Verbindungsstatus

Sie können die Verbindungsintegritätskonfiguration für die AWS CRT-basierten HTTP-Clients konfigurieren, indem Sie die `connectionHealthConfiguration` Methode im HTTP-Client-Builder verwenden.

Im folgenden Beispiel wird ein S3-Dienstclient erstellt, der eine AWS CRT-basierte HTTP-Clientinstanz verwendet, die mit einer Verbindungsintegritätskonfiguration und einer maximalen Leerlaufzeit für Verbindungen konfiguriert ist.

### Synchronous client

#### Importe

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

#### Code

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

```
        .connectionMaxIdleTime(Duration.ofSeconds(5)))
        .build();

// Perform work with s3Client.

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

## Asynchronous client

### Importe

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

### Code

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

// Perform work with s3AsyncClient.

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

## HTTP/2-Support

Das HTTP/2-Protokoll wird in den AWS CRT-basierten HTTP-Clients noch nicht unterstützt, ist aber für eine future Version geplant.

Wenn Sie in der Zwischenzeit Service-Clients verwenden, die HTTP/2-Unterstützung benötigen, wie z. B. der [KinesisAsyncClient](#) oder der, sollten Sie in Erwägung ziehen [TranscribeStreamingAsyncClient](#), stattdessen den zu verwenden. [NettyNioAsyncHttpClient](#)

## Beispiel für eine Proxykonfiguration

Der folgende Codeausschnitt zeigt die Verwendung der Einstellung [ProxyConfiguration.Builder](#), die Sie zur Konfiguration der Proxyeinstellungen verwenden, im Code.

### Synchronous client

#### Importe

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

#### Code

```
SdkHttpClient crtHttpClient = AwsCrtHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .scheme("https")
        .host("myproxy")
        .port(1234)
        .username("username")
        .password("password")
        .build())
    .build();
```

### Asynchronous client

#### Importe

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

#### Code

```
SdkAsyncHttpClient crtAsyncHttpClient = AwsCrtAsyncHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .scheme("https")
        .host("myproxy")
        .port(1234)
        .username("username")
```

```
.password("password")
    .build())
    .build();
```

Die entsprechenden Java-Systemeigenschaften für die Proxykonfiguration werden im folgenden Befehlszeilenausschnitt angezeigt.

```
$ java -Dhttps.proxyHost=myproxy -Dhttps.proxyPort=1234 -Dhttps.proxyUser=username \
-Dhttps.proxyPassword=password -cp ... App
```

### Important

Um eine der HTTPS-Proxy-Systemeigenschaften verwenden zu können, muss die `scheme` Eigenschaft im Code auf gesetzt werden. `https` Wenn die Schemaeigenschaft nicht im Code festgelegt ist, verwendet das Schema standardmäßig HTTP und das SDK sucht nur nach `http.*` Systemeigenschaften.

## Ausnahmebehandlung für die AWS SDK for Java 2.x

Damit Sie hochwertige Anwendungen mit dem SDK erstellen können, sollten Sie unbedingt verstehen, wie und wann das AWS SDK for Java 2.x Ausnahmen auslöst. In den folgenden Abschnitten werden die verschiedenen Fälle von Ausnahmen beschrieben, die vom SDK ausgelöst werden, und wie sie korrekt verarbeitet werden.

### Warum nicht aktivierte Ausnahmen?

Das AWS SDK for Java verwendet Laufzeitausnahmen (bzw. ungeprüfte Ausnahmen) anstelle geprüfter Ausnahmen, und zwar aus folgenden Gründen:

- Entwickler erhalten genaue Kontrolle über die Fehler, auf die sie eingehen möchten. Sie werden aber nicht dazu gezwungen, auftretende Ausnahmen zu verarbeiten, für die sie sich nicht interessieren (was den Code übermäßig aufblähen würde).
- Skalierbarkeitsprobleme durch geprüfte Ausnahmen in großen Anwendungen werden verhindert.

Im Allgemeinen eignen sich geprüfte Ausnahmen gut im kleinen Rahmen. Wenn Anwendungen wachsen und komplexer werden, können sie allerdings zu Problemen führen.

## AwsServiceException (und Unterklassen)

[AwsServiceException](#) ist die häufigste Ausnahme, die bei der Verwendung von auftrittAWS SDK for Java. `AwsServiceException` ist eine Unterklasse der allgemeineren [SdkServiceException](#). `AwsServiceExceptions` stellen eine Fehlerantwort von einem darAWS-Service. Wenn Sie beispielsweise versuchen, eine Instance zu beenden, Amazon EC2 die nicht existiert, Amazon EC2 gibt eine Fehlerantwort zurück und alle Details dieser Fehlerantwort werden in die `AwsServiceException` ausgelöste aufgenommen.

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

`AwsServiceException` gibt Ihnen Informationen wie z. B.:

- zurückgegebener HTTP-Statuscode
- Zurückgegebener AWS Fehlercode
- Detaillierte Fehlermeldung vom Service in der [AwsErrorDetails](#) Klasse
- AWS -Anforderungs-ID für die fehlgeschlagene Anforderung

In den meisten Fällen `AwsServiceException` wird eine servicespezifische Unterklasse von ausgelöst, um Entwicklern eine differenzierte Kontrolle über die Behandlung von Fehlerfällen durch Fangblöcke zu ermöglichen. Die Java-SDK-API-Referenz für [AwsServiceException](#) zeigt die große Anzahl von `AwsServiceException` Unterklassen an. Verwenden Sie die Unterklassenlinks, um eine Aufschlüsselung durchzuführen, um die detaillierten Ausnahmen anzuzeigen, die von einem Service ausgelöst werden.

Die folgenden Links zur SDK-API-Referenz zeigen beispielsweise die Ausnahmehierarchien für einige gängige AWS-Services. Die Liste der Unterklassen auf den einzelnen Seiten zeigt die spezifischen Ausnahmen, die Ihr Code abfangen kann.

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

Um mehr über eine Ausnahme zu erfahren, überprüfen Sie die `errorCode` auf dem [-AwsErrorDetails](#) Objekt. Sie können den `-errorCode` Wert verwenden, um Informationen in der

Service-Leitfaden-API nachzuschlagen. Wenn beispielsweise ein abgefangen `S3Exception` wird und der `AwsErrorDetails#errorCode()` Wert ist `InvalidRequest`, verwenden Sie die [Liste der Fehlercodes](#) in der Amazon S3-API-Referenz, um weitere Details anzuzeigen.

## SdkClientException

[SdkClientException](#) zeigt an, dass im Java-Clientcode ein Problem aufgetreten ist, entweder beim Versuch, eine Anfrage an zu senden, AWS oder beim Versuch, eine Antwort von zu analysieren AWS. Ein `SdkClientException` ist im Allgemeinen schwerwiegender als ein `SdkServiceException` und weist auf ein großes Problem hin, das den Client daran hindert, Service-Aufrufe an `-AWSServices` durchzuführen. Beispielsweise löst das AWS SDK for Java eine `SdkClientException` aus, wenn zum Zeitpunkt des Versuchs, eine Operation auf einem der Clients durchzuführen, keine Netzwerkverbindung verfügbar ist.

## Ausnahmen und Wiederholungsverhalten

Das SDK for Java wiederholt Anforderungen für mehrere [clientseitige Ausnahmen](#) und für [HTTP-Statuscodes](#), die es von AWS-Service Antworten erhält. Diese Fehler werden als Teil des veralteten `RetryMode`, das Service-Clients standardmäßig verwenden. Die Java-API-Referenz für [RetryMode](#) beschreibt die verschiedenen Möglichkeiten, wie Sie den Modus konfigurieren können.

Um die Ausnahmen und HTTP-Statuscodes anzupassen, die automatische Wiederholungen auslösen, konfigurieren Sie Ihren Service-Client mit einem [RetryPolicy](#), der - [RetryOnExceptionsCondition](#) und [-RetryOnStatusCodeCondition](#) Instances hinzufügt.

## Protokollierung mit dem SDK for Java 2.x

Die AWS SDK for Java 2.x verwendet [SLF4J](#), eine Abstraktionsebene, die die Verwendung eines von mehreren Protokollierungssystemen zur Laufzeit ermöglicht.

Zu den unterstützten Protokollierungssystemen gehören unter anderem das Java Logging Framework und Apache [Log4j 2](#). In diesem Thema erfahren Sie, wie Sie Log4j 2 als Protokollierungssystem für die Arbeit mit dem SDK verwenden.

## Log4j 2-Konfigurationsdatei

In der Regel verwenden Sie eine Konfigurationsdatei mit dem Namen `log4j2.xml` Log4j 2. Beispiel-Konfigurationsdateien werden nachfolgend angezeigt. Weitere Informationen über die Werte in der Konfigurationsdatei finden Sie im [Handbuch für die Log4j-Konfiguration](#).

Die `log4j2.xml` Datei muss sich beim Start Ihrer Anwendung im Klassenpfad befinden. Legen Sie für ein Maven-Projekt die Datei im `<project-dir>/src/main/resources` Verzeichnis ab.

Die `log4j2.xml` Konfigurationsdatei gibt Eigenschaften wie die [Protokollierungsebene an](#), an die die Protokollierungsausgabe gesendet wird (z. B. [an eine Datei oder an die Konsole](#)) und das [Format der Ausgabe](#). Die Protokollierungsebene gibt den Detaillierungsgrad an, den Log4j 2 ausgibt. Log4j 2 unterstützt das Konzept mehrerer [Protokollierungshierarchien](#). Die Protokollierungsebene wird für jede Hierarchie separat festgelegt. Die Hauptprotokollierungshierarchie, die Sie mit der verwenden, AWS SDK for Java 2.x ist `software.amazon.awssdk`.

## Protokollierungsabhängigkeit hinzufügen

Verwenden Sie Folgendes, um die Log4j-2-Bindung für SLF4J in Ihrer Build-Datei zu konfigurieren.

### Maven

Fügen Sie Ihrer `pom.xml` Datei die folgenden Elemente hinzu.

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

### Gradle–Kotlin DSL

Fügen Sie Ihrer `build.gradle.kts` Datei Folgendes hinzu.

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

Verwenden Sie `2.20.0` für die Mindestversion des `log4j-slf4j2-impl` Artefakts. Verwenden Sie für die neueste Version die in [Maven Central veröffentlichte](#) Version. Ersetzen Sie `VERSION` durch die Version, die Sie verwenden werden.



## SDK-spezifische Fehler und Warnungen

Wir empfehlen Ihnen, die Logger-Hierarchie „software.amazon.awssdk“ immer auf „WARN“ eingestellt zu lassen, um wichtige Nachrichten aus den Clientbibliotheken des SDK abzufangen. Wenn der Amazon S3-Client beispielsweise erkennt, dass Ihre Anwendung ein nicht ordnungsgemäß geschlossen hat `InputStream` und Ressourcen preisgeben könnte, meldet der S3-Client es über eine Warnmeldung an die Protokolle. Dadurch wird auch sichergestellt, dass Nachrichten protokolliert werden, wenn der Client Schwierigkeiten bei der Verarbeitung von Anforderungen oder Antworten hat.

Die folgende `log4j2.xml` Datei setzt `rootLogger` auf „WARN“, was dazu führt, dass Nachrichten auf Warn- und Fehlerebene von allen Loggern in der Anwendung ausgegeben werden, einschließlich solcher in der Hierarchie „software.amazon.awssdk“. Alternativ können Sie die Logger-Hierarchie „software.amazon.awssdk“ explizit auf „WARN“ setzen, wenn verwendet `<Root level="ERROR">` wird.

### Beispiel für eine `Log4j2.xml`-Konfigurationsdatei

Diese Konfiguration protokolliert Nachrichten auf den Ebenen „FEHLER“ und „WARN“ in der Konsole für alle Logger-Hierarchien.

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

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

## Protokollierung der Anforderungs-/Antwortübersicht

Jede Anforderung an einen AWS-Service generiert eine eindeutige AWS Anforderungs-ID, die nützlich ist, wenn ein Problem damit auftritt, wie ein eine Anforderung AWS-Service verarbeitet. AWS Anforderungs-IDs sind programmgesteuert über [SdkServiceException](#) Objekte im SDK für

fehlgeschlagene Service-Aufrufe zugänglich und können auch über die „DEBUG“-Protokollebene des Loggers „software.amazon.awssdk.request“ gemeldet werden.

Die folgende `log4j2.xml` Datei ermöglicht eine Zusammenfassung der Anforderungen und Antworten.

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

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

Hier finden Sie ein Beispiel für die Protokollausgabe:

```
2022-09-23 16:02:08 [main] DEBUG software.amazon.awssdk.request:85 - Sending Request:
DefaultSdkHttpRequest(httpMethod=POST, protocol=https, host=dynamodb.us-
east-1.amazonaws.com, encodedPath=/, headers=[amz-sdk-invocation-id, Content-Length,
Content-Type, User-Agent, X-Amz-Target], queryParameters=[])
2022-09-23 16:02:08 [main] DEBUG software.amazon.awssdk.request:85 - Received
successful response: 200, Request ID:
QS9DUMME2NHEDH8TGT9N5V530JV4KQNS05AEMVJF66Q9ASUAAJG, Extended Request ID: not
available
```

Wenn Sie nur an der Anforderungs-ID interessiert sind, verwenden Sie `<Logger name="software.amazon.awssdk.requestId" level="DEBUG" />`.

## Verbose-Wire-Protokollierung

Es kann nützlich sein, die genauen Anfragen und Antworten zu sehen, die das SDK for Java 2.x sendet und empfängt. Wenn Sie Zugriff auf diese Informationen benötigen, können Sie sie vorübergehend aktivieren, indem Sie die erforderliche Konfiguration je nach HTTP-Client hinzufügen, den der Service-Client verwendet.

Standardmäßig verwenden synchrone Service-Clients wie der [S3Client](#) einen zugrunde liegenden Apache HttpClient und asynchrone Service-Clients wie den [S3AsyncClient](#) einen nicht blockierenden Netty-HTTP-Client.

Hier finden Sie eine Aufschlüsselung der HTTP-Clients, die Sie für die beiden Kategorien von Service-Clients verwenden können:

Synchrone HTTP-Clients	Asynchrone HTTP-Clients
<a href="#">ApacheHttpClient</a> (Standard)	<a href="#">NettyNioAsyncHttpClient</a> (Standard)
<a href="#">URLConnectionHttpClient</a>	<a href="#">AwsCrtAsyncHttpClient</a>

Konfigurationseinstellungen, die Sie je nach zugrundeliegendem HTTP-Client hinzufügen müssen, finden Sie auf der entsprechenden Registerkarte unten.

#### Warning

Wir empfehlen Ihnen, die Übertragungsprotokollierung ausschließlich für Debugging-Zwecke zu verwenden. Deaktivieren Sie sie in Produktionsumgebungen, da sie sensible Daten aufzeichnen kann. Sie protokolliert die gesamte Anfrage oder Antwort ohne Verschlüsselung, auch bei einem HTTPS-Aufruf. Bei großen Anforderungen (z. B. zum Hochladen einer Datei auf Amazon S3) oder Antworten kann die ausführliche Protokollierung auch die Leistung Ihrer Anwendung erheblich beeinträchtigen.

## ApacheHttpClient

Fügen Sie der `log4j2.xml` Konfigurationsdatei den Logger „org.apache.http.Speed“ hinzu und setzen Sie die Stufe auf „DEBUG“.

Die folgende `log4j2.xml` Datei aktiviert die vollständige Wire-Protokollierung für den Apache HttpClient.

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

```

</Appenders>

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

```

Eine zusätzliche Maven-Abhängigkeit vom `log4j-1.2-api` Artefakt ist für die Wire-Protokollierung mit Apache erforderlich, da 1.2 unter der Hintergrundumgebung verwendet wird.

Die vollständigen Maven-Abhängigkeiten für `log4j 2`, einschließlich der Wire-Protokollierung für den Apache-HTTP-Client, werden in den folgenden Build-Dateiausschnitten angezeigt.

## Maven

```

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

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

```

```
<artifactId>log4j-1.2-api</artifactId>
</dependency>
...
```

## Gradle-Kotlin-DSL

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

Verwenden Sie `2.20.0` für die Mindestversion des `log4j-bom` Artefakts. Verwenden Sie für die neueste Version die in [Maven Central veröffentlichte](#) Version. Ersetzen Sie `VERSION` durch die Version, die Sie verwenden werden.

## URLConnectionHttpClient

Um Details für Service-Clients zu protokollieren, die die `URLConnectionHttpClient` verwenden, erstellen Sie zunächst eine `logging.properties` Datei mit dem folgenden Inhalt:

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

Legen Sie die folgende JVM-Systemeigenschaft mit dem vollständigen Pfad des `festlogging.properties`:

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

Diese Konfiguration protokolliert nur die Header der Anforderung und Antwort, zum Beispiel:

```
<Request> FINE: sun.net.www.MessageHeader@35a9782c11 pairs: {GET /fileuploadtest
HTTP/1.1: null}{amz-sdk-invocation-id: 5f7e707e-4ac5-bef5-ba62-00d71034ffdc}
{amz-sdk-request: attempt=1; max=4}{Authorization: AWS4-HMAC-SHA256
Credential=<deleted>/20220927/us-east-1/s3/aws4_request, SignedHeaders=amz-sdk-
invocation-id;amz-sdk-request;host;x-amz-content-sha256;x-amz-date;x-amz-te,
```

```
Signature=e367fa0bc217a6a65675bb743e1280cf12f8e8d566196a816d948fdf0b42ca1a){User-Agent: aws-sdk-java/2.17.230 Mac_OS_X/12.5 OpenJDK_64-Bit_Server_VM/25.332-b08 Java/1.8.0_332 vendor/Amazon.com_Inc. io/sync http/URLConnection cfg/retry-mode/legacy}{x-amz-content-sha256: UNSIGNED-PAYLOAD}{X-Amz-Date: 20220927T133955Z}{x-amz-
te: append-md5}{Host: tkhill-test1.s3.amazonaws.com}{Accept: text/html, image/gif,
image/jpeg, *, q=.2, */*; q=.2}{Connection: keep-alive}
<Response> FINE: sun.net.www.MessageHeader@70a36a6611 pairs: {null: HTTP/1.1
200 OK}{x-amz-id-2: sAFeZD0KdUMsBbkDjyDZw7P0oocb4C9KbiuzfJ6TWKQsGXHM/
dFu0vr2tUb7Y1wEHGdJ3DSIxq0=}{x-amz-request-id: P9QW9SMZ97FKZ9X7}{Date: Tue,
27 Sep 2022 13:39:57 GMT}{Last-Modified: Tue, 13 Sep 2022 14:38:12 GMT}{ETag:
"2cbe5ad4a064cedec33b452bebf48032"}{x-amz-transfer-encoding: append-md5}{Accept-
Ranges: bytes}{Content-Type: text/plain}{Server: AmazonS3}{Content-Length: 67}
```

Um die Anforderungs-/Antworttexte anzuzeigen, fügen Sie `-Djavax.net.debug=all` zu den JVM-Eigenschaften hinzu. Diese zusätzliche Eigenschaft protokolliert eine Vielzahl von Informationen, einschließlich aller SSL-Informationen.

Suchen Sie in der Protokollkonsole oder der Protokolldatei nach "GET" oder , "POST" um schnell zum Abschnitt des Protokolls zu gelangen, der tatsächliche Anforderungen und Antworten enthält. Suchen Sie "Plaintext before ENCRYPTION" nach Anforderungen und "Plaintext after DECRYPTION" Antworten, um den vollständigen Text der Header und Textkörper anzuzeigen.

## NettyNioAsyncHttpClient

Wenn Ihr asynchroner Service-Client den Standard- verwendet `NettyNioAsyncHttpClient`, fügen Sie Ihrer `log4j2.xml` Datei zwei zusätzliche Logger hinzu, um HTTP-Header und Anforderungs-/Antworttexte zu protokollieren.

```
<Logger name="io.netty.handler.logging" level="DEBUG" />
<Logger name="io.netty.handler.codec.http2.Http2FrameLogger" level="DEBUG" />
```

Hier ist ein vollständiges `log4j2.xml` Beispiel:

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

```

<Loggers>
  <Root level="WARN">
    <AppenderRef ref="ConsoleAppender"/>
  </Root>
  <Logger name="software.amazon.awssdk" level="WARN" />
  <Logger name="software.amazon.awssdk.request" level="DEBUG" />
  <Logger name="io.netty.handler.logging" level="DEBUG" />
  <Logger name="io.netty.handler.codec.http2.Http2FrameLogger" level="DEBUG" /
>
</Loggers>
</Configuration>

```

Diese Einstellungen protokollieren alle Header-Details und Anforderungs-/Antworttexte.

## AwsCrtAsyncHttpClient

Wenn Sie Ihren Service-Client für die Verwendung einer Instance von konfiguriert haben `AwsCrtAsyncHttpClient`, können Sie Details protokollieren, indem Sie die JVM-Systemeigenschaften oder programmgesteuert festlegen.

### Log to a file at "Debug" level

Verwenden von Systemeigenschaften:

```

-Daws.crt.log.level=Trace
-Daws.crt.log.destination=File
-Daws.crt.log.filename=<path to file>

```

Programmgesteuert:

```

import software.amazon.awssdk.crt.Log;

// Execute this statement before constructing the
// SDK service client.
Log.initLoggingToFile(Log.LogLevel.Trace,
"<path to file>");

```

### Log to the console at "Debug" level

Verwenden von Systemeigenschaften:

```

-Daws.crt.log.level=Trace
-Daws.crt.log.destination=Stdout

```

Programmgesteuert:

```

import software.amazon.awssdk.crt.Log;

// Execute this statement before constructing the
// SDK service client.
Log.initLoggingToStdout(Log.LogLevel.Trace);

```

Aus Sicherheitsgründen protokolliert `AwsCrtAsyncHttpClient` auf der Ebene „Trace“ nur Antwort-Header. Anforderungs-Header, Anforderungstexte und Antworttexte werden nicht protokolliert.

## Festlegen der JVM-TTL für DNS-Namenssuche

Die Java Virtual Machine (JVM) speichert DNS-Namensauflösungen zwischen. Wenn die JVM einen Hostnamen in eine IP-Adresse auflöst, speichert sie die IP-Adresse für einen bestimmten Zeitraum zwischen, bekannt als time-to-live (TTL).

Da AWS Ressourcen DNS-Namenseinträge verwenden, die sich gelegentlich ändern, empfehlen wir Ihnen, Ihre JVM mit einem TTL-Wert von nicht mehr als 60 Sekunden zu konfigurieren. Auf diese Weise wird bei Änderung der IP-Adresse einer Ressource sichergestellt, dass Ihre Anwendung die neue IP-Adresse der Ressource durch erneute Abfrage des DNS abrufen und nutzen kann.

Bei einigen Java-Konfigurationen ist die JVM-Standard-TTL so festgelegt, dass DNS-Einträge nie aktualisiert werden, bis die JVM neu gestartet wird. Wenn sich die IP-Adresse für eine - AWS Ressource ändert, während Ihre Anwendung noch ausgeführt wird, kann sie diese Ressource erst verwenden, wenn Sie die JVM manuell neu starten und die zwischengespeicherten IP-Informationen aktualisiert werden. In diesem Fall ist es wichtig, die TTL der JVM so einzustellen, dass sie die zwischengespeicherten IP-Daten von Zeit zu Zeit aktualisiert.

### Note

Die Standard-TTL kann je nach Version Ihrer JVM und abhängig davon, ob ein [Sicherheits-Manager](#) installiert ist, unterschiedlich sein. Viele JVMs bieten eine Standard-TTL von weniger als 60 Sekunden. Wenn Sie eine solche JVM und keinen Sicherheits-Manager nutzen, können Sie den Rest dieses Themas ignorieren.

## Festlegen der JVM-TTL

Sie können die TTL der JVM ändern, indem Sie den Eigenschaftswert [networkaddress.cache.ttl](#) festlegen. Nutzen Sie dazu eine der folgenden Methoden je nach Ihrem Bedarf:

- global für alle Anwendungen, die die JVM nutzen. Legen Sie `networkaddress.cache.ttl` in der Datei `$JAVA_HOME/jre/lib/security/java.security` fest:



```
networkaddress.cache.ttl=60
```

- nur für Ihre Anwendung, legen Sie `networkaddress.cache.ttl` im Initialisierungscode Ihrer Anwendung fest:

```
java.security.Security.setProperty("networkaddress.cache.ttl" , "60");
```

## Bewährte Methoden für AWS SDK for Java 2.x

In diesem Abschnitt werden bewährte Methoden für die Verwendung des SDK for Java 2.x aufgeführt.

### Themen

- [Verwenden Sie nach Möglichkeit einen SDK-Client erneut](#)
- [Schließt Eingabestreams aus Client-Vorgängen](#)
- [Optimieren Sie HTTP-Konfigurationen auf der Grundlage von Leistungstests](#)
- [Verwenden Sie OpenSSL für den Netty-basierten HTTP-Client](#)
- [API-Timeouts konfigurieren](#)
- [Verwenden Sie Metriken](#)

## Verwenden Sie nach Möglichkeit einen SDK-Client erneut

Jeder SDK-Client unterhält seinen eigenen HTTP-Verbindungspool. Eine Verbindung, die bereits im Pool vorhanden ist, kann durch eine neue Anfrage wiederverwendet werden, um die Zeit für den Aufbau einer neuen Verbindung zu verkürzen. Wir empfehlen, eine einzelne Instanz des Clients gemeinsam zu nutzen, um den Mehraufwand zu vermeiden, der durch zu viele Verbindungspools entsteht, die nicht effektiv genutzt werden. Alle SDK-Clients sind threadsicher.

Wenn Sie eine Client-Instanz nicht gemeinsam nutzen möchten, rufen Sie die Instanz `close()` auf, um die Ressourcen freizugeben, wenn der Client nicht benötigt wird.

## Schließt Eingabestreams aus Client-Vorgängen

Für Streaming-Operationen [S3Client#getObject](#), z. B. wenn Sie [ResponseInputStream](#) direkt mit arbeiten, empfehlen wir Ihnen, wie folgt vorzugehen:

- Lesen Sie so schnell wie möglich alle Daten aus dem Eingabestream.
- Schließen Sie den Eingabestream so schnell wie möglich.

Wir geben diese Empfehlungen ab, weil der Eingabestream ein direkter Datenstrom aus der HTTP-Verbindung ist und die zugrunde liegende HTTP-Verbindung erst wiederverwendet werden kann, wenn alle Daten aus dem Stream gelesen und der Stream geschlossen wurde. Wenn diese Regeln nicht befolgt werden, können dem Client die Ressourcen ausgehen, indem zu viele offene, aber ungenutzte HTTP-Verbindungen zugewiesen werden.

## Optimieren Sie HTTP-Konfigurationen auf der Grundlage von Leistungstests

Das SDK bietet eine Reihe von [Standard-HTTP-Konfigurationen](#), die für allgemeine Anwendungsfälle gelten. Wir empfehlen Kunden, die HTTP-Konfigurationen für ihre Anwendungen auf der Grundlage ihrer Anwendungsfälle zu optimieren.

Als guten Ausgangspunkt bietet das SDK eine Funktion für [intelligente Standardkonfigurationen](#). Diese Funktion ist ab Version 2.17.102 verfügbar. Sie wählen je nach Anwendungsfall einen Modus, der sinnvolle Konfigurationswerte bietet.

## Verwenden Sie OpenSSL für den Netty-basierten HTTP-Client

Standardmäßig verwenden die SDKs [NettyNioAsyncHttpClient](#) die Standard-SSL-Implementierung des JDK als `SslProvider`. Unsere Tests haben ergeben, dass OpenSSL besser abschneidet als die Standardimplementierung von JDK. Die Netty-Community [empfiehlt außerdem die Verwendung von OpenSSL](#).

Um OpenSSL zu verwenden, fügen Sie Ihre Abhängigkeiten `netty-tcnative` hinzu. Einzelheiten zur Konfiguration finden Sie in der [Netty-Projektdokumentation](#).

Nachdem Sie für Ihr Projekt `netty-tcnative` konfiguriert haben, wählt die `NettyNioAsyncHttpClient` Instanz automatisch OpenSSL aus. Alternativ können Sie das `SslProvider` explizit mithilfe des `NettyNioAsyncHttpClient` Builders festlegen, wie im folgenden Snippet gezeigt.

```
NettyNioAsyncHttpClient.builder()
    .sslProvider(SslProvider.OPENSSL)
    .build();
```

## API-Timeouts konfigurieren

Das SDK bietet [Standardwerte](#) für einige Timeout-Optionen wie Verbindungs-Timeout und Socket-Timeouts, jedoch nicht für Timeouts bei API-Aufrufen oder Timeouts für einzelne API-Aufrufversuche. Es empfiehlt sich, Timeouts sowohl für einzelne Versuche als auch für die gesamte Anfrage festzulegen. Auf diese Weise wird sichergestellt, dass Ihre Anwendung schnell und optimal fehlschlägt, wenn vorübergehende Probleme auftreten, die dazu führen können, dass Anforderungsversuche länger dauern können, oder bei schwerwiegenden Netzwerkproblemen.

Sie können Timeouts für alle Anfragen konfigurieren, die von einem Service-Client gestellt werden, indem Sie [ClientOverrideConfiguration#apiCallAttemptTimeout](#) und [ClientOverrideConfiguration#apiCallTimeout](#) verwenden.

Das folgende Beispiel zeigt die Konfiguration eines Amazon S3 S3-Clients mit benutzerdefinierten Timeout-Werten.

```
S3Client.builder()
    .overrideConfiguration(
        b -> b.apiCallTimeout(Duration.ofSeconds(<custom value>))
            .apiCallAttemptTimeout(Duration.ofMillis(<custom value>)))
    .build();
```

### **apiCallAttemptTimeout**

Diese Einstellung legt die Zeitspanne für einen einzelnen HTTP-Versuch fest, nach deren Ablauf der API-Aufruf erneut versucht werden kann.

### **apiCallTimeout**

Der Wert für diese Eigenschaft konfiguriert die Zeitspanne für die gesamte Ausführung, einschließlich aller Wiederholungsversuche.

Als Alternative zur Einstellung dieser Timeout-Werte auf dem Service-Client können Sie eine einzelne Anforderung verwenden [RequestOverrideConfiguration#apiCallTimeout\(\)](#) und [RequestOverrideConfiguration#apiCallAttemptTimeout\(\)](#) konfigurieren.

Im folgenden Beispiel wird eine einzelne `listBuckets` Anfrage mit benutzerdefinierten Timeout-Werten konfiguriert.

```
s3Client.listBuckets(lbr -> lbr.overrideConfiguration(
```

```
b -> b.apiCallTimeout(Duration.ofSeconds(<custom value>))
    .apiCallAttemptTimeout(Duration.ofMillis(<custom value>))));
```

Wenn Sie diese Eigenschaften zusammen verwenden, legen Sie ein festes Limit für die Gesamtzeit fest, die für alle Versuche bei Wiederholungsversuchen aufgewendet wird. Sie legen außerdem fest, dass eine einzelne HTTP-Anfrage bei einer langsamen Anfrage schnell fehlschlägt.

## Verwenden Sie Metriken

Das SDK for Java kann [Metriken für die Service-Clients in Ihrer Anwendung sammeln](#). Sie können diese Messwerte verwenden, um Leistungsprobleme zu identifizieren, allgemeine Nutzungstrends zu überprüfen, zurückgemeldete Service-Client-Ausnahmen zu überprüfen oder um ein bestimmtes Problem genauer zu untersuchen.

Wir empfehlen Ihnen, Metriken zu sammeln und anschließend die CloudWatch Amazon-Logs zu analysieren, um ein tieferes Verständnis der Leistung Ihrer Anwendung zu erhalten.

# Verwenden der Funktionen von AWS SDK for Java 2.x

## Allgemeine Funktionen

Das SDK for Java 2.x enthält mehrere Funktionen, mit denen die Programmierung von AWS-Services vereinfacht wird.

- Das SDK blendet die komplexen Mechanismen für das [Abrufen paginierter Ergebnisse](#) und das [Abfragen von Ressourcen aus](#).
- [Die asynchrone Programmierung mit nicht blockierenden E/A-Vorgängen](#) hilft Ihnen, gleichzeitigen Code mit besserer Leistung zu schreiben. Das SDK bietet nach Möglichkeit die Vorteile von [HTTP/2](#), z. B. eine geringere Latenz.
- Das Java SDK kann [Metriken](#) generieren, die Ihnen helfen, den Betriebszustand Ihrer Anwendungen zu überwachen.

## Servicespezifische Funktionen

Zusätzlich zu den zuvor erwähnten allgemeinen Funktionen bietet das Java SDK Funktionen für bestimmte AWS-Services.

- Amazon S3 – Um [Ihre Arbeit mit Dateien und Verzeichnissen mit Amazon S3 zu vereinfachen](#), stellt das SDK den S3 Transfer Manager bereit. Amazon S3 Um die [Leistung und Zuverlässigkeit bei Verwendung der asynchronen Standard-S3-API des SDK zu verbessern](#), bietet das SDK den AWS CRT-basierten S3-Client an. S3
- DynamoDB – [Objektorientierte Zuordnungsfunktionen](#) werden von der DynamoDB Enhanced Client API bereitgestellt. [Arbeiten Sie mit dokumentorientierten Daten im JSON-Stil](#), indem Sie die Enhanced Document API verwenden.
- IAM – Die IAM Policy Builder API bietet eine [objektorientierte, typsichere Möglichkeit, IAM-Richtlinien zu erstellen](#).

# Arbeiten mit paginierten Ergebnissen mithilfe von AWS SDK for Java 2.x

Viele AWS Operationen geben paginierte Ergebnisse zurück, wenn das Antwortobjekt zu groß ist, um es in einer einzigen Antwort zurückzugeben. In der AWS SDK for Java 1.0 enthält die Antwort ein Token, mit dem Sie die nächste Seite mit Ergebnissen abrufen. Im Gegensatz dazu verfügt der AWS SDK for Java 2.x über Autopaginierungsmethoden, die mehrere Serviceaufrufe durchführen, um die nächste Seite mit Ergebnissen automatisch für Sie zu erhalten. Sie müssen nur Code schreiben, der die Ergebnisse verarbeitet. Die Autopaginierung ist sowohl für synchrone als auch für asynchrone Clients verfügbar.

## Note

Diese Codeausschnitte setzen voraus, dass Sie [die Grundlagen der Verwendung des SDK](#) verstehen und Ihre Umgebung mit [Single-Sign-On-Zugriff](#) konfiguriert haben.

## Synchrone Paginierung

Die folgenden Beispiele zeigen synchrone Paginierungsmethoden zum Auflisten von Objekten in einem - Amazon S3 Bucket.

### Über Seiten iterieren

Das erste Beispiel zeigt die Verwendung eines Paginatorobjekts, einer `ListRes Instance` [ListObjectsV2Iterable](#), um mit der `stream` Methode alle Antwortseiten zu durchlaufen. Der Code streamt über die Antwortseiten, konvertiert den Antwortstream in einen [S3Object](#) Inhaltsstream und verarbeitet dann den Inhalt des Amazon S3 Objekts.

Die folgenden Importe gelten für alle Beispiele in diesem Abschnitt zur synchronen Paginierung.

### Importe

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
```

```

import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;

```

```

ListObjectsV2Request listReq = ListObjectsV2Request.builder()
    .bucket(bucketName)
    .maxKeys(1)
    .build();

ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
// Process response pages
listRes.stream()
    .flatMap(r -> r.contents().stream())
    .forEach(content -> System.out
        .println(" Key: " + content.key() + "
size = " + content.size()));

```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Iterieren über Objekte

Die folgenden Beispiele zeigen Möglichkeit, um über die in der Antwort zurückgegebenen Objekte anstatt über die Seiten der Antwort zu iterieren. Die `-contents` Methode der

-`ListObjectsV2Iterable` Klasse gibt einen zurück `SdkIterable`, der mehrere Methoden zur Verarbeitung der zugrunde liegenden Inhaltselemente bereitstellt.

## Verwenden eines Streams

Der folgende Codeausschnitt verwendet die `-stream` Methode für den Antwortinhalt, um über die paginierte Elementauflistung zu iterieren.

```
// Helper method to work with paginated collection of items directly.
listRes.contents().stream()
    .forEach(content -> System.out
        .println(" Key: " + content.key() + "
size = " + content.size()));
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Verwenden einer für jede Schleife

Da die `Iterable` Schnittstelle `SdkIterable` erweitert, können Sie den Inhalt wie jeden `Iterable` verarbeiten. Der folgende Codeausschnitt verwendet eine `for-each` Standardschleife, um den Inhalt der Antwort zu durchlaufen.

```
for (S3Object content : listRes.contents()) {
    System.out.println(" Key: " + content.key() + " size = " +
content.size());
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Manuelle Paginierung

Wenn Ihr Anwendungsfall es erfordert, ist die manuelle Paginierung weiterhin verfügbar. Verwenden Sie das nächste Token im Antwortobjekte für die nachfolgenden Anforderungen. Im folgenden Beispiel wird eine `while` Schleife verwendet.

```
ListObjectsV2Request listObjectsReqManual =
ListObjectsV2Request.builder()
    .bucket(bucketName)
    .maxKeys(1)
```



```

        .build();

        boolean done = false;
        while (!done) {
            ListObjectsV2Response listObjResponse =
s3.listObjectsV2(listObjectsReqManual);
            for (S3Object content : listObjResponse.contents()) {
                System.out.println(content.key());
            }

            if (listObjResponse.nextContinuationToken() == null) {
                done = true;
            }

            listObjectsReqManual = listObjectsReqManual.toBuilder()

                .continuationToken(listObjResponse.nextContinuationToken())

                    .build();
        }
    }
}

```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Asynchrone Paginierung

Die folgenden Beispiele zeigen asynchrone Paginierungsmethoden zum Auflisten von DynamoDB Tabellen.

### Iterieren über Seiten mit Tabellennamen

In den folgenden beiden Beispielen wird ein asynchroner DynamoDB-Client verwendet, der die `-listTablesPaginator` Methode mit einer `-Anforderung` aufruft, um eine zu erhalten [ListTablesPublisher](#). `ListTablesPublisher` implementiert zwei Schnittstellen, die viele Optionen für die Verarbeitung von Antworten bieten. Wir werden uns die Methoden der einzelnen Schnittstellen ansehen.

#### Verwenden eines **Subscriber**

Das folgende Codebeispiel zeigt, wie paginierte Ergebnisse mithilfe der von implementierten `org.reactivestreams.Publisher` Schnittstelle verarbeitet werden `ListTablesPublisher`. Weitere Informationen zum Modell reaktiver Streams finden Sie im [GitHub Repo](#).

Die folgenden Importe gelten für alle Beispiele in diesem Abschnitt zur asynchronen Paginierung.

## Importe

```
import io.reactivex.rxjava3.core.Flowable;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import reactor.core.publisher.Flux;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import software.amazon.awssdk.services.dynamodb.paginators.ListTablesPublisher;

import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
```

Der folgende Code ruft eine `ListTablesPublisher` Instance ab.

```
// Creates a default client with credentials and region loaded from the
// environment.
final DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();

ListTablesRequest listTablesRequest =
ListTablesRequest.builder().limit(3).build();
ListTablesPublisher publisher =
asyncClient.listTablesPaginator(listTablesRequest);
```

Der folgende Code verwendet eine anonyme Implementierung von `org.reactivestreams.Subscriber`, um die Ergebnisse für jede Seite zu verarbeiten.

Die `onSubscribe`-Methode ruft die `Subscription.request`-Methode auf, um Anforderungen von Daten vom Publisher zu initiieren. Diese Methode muss aufgerufen werden, um den Abruf von Daten vom Publisher zu starten.

Die `-onNext`-Methode des Abonnenten verarbeitet eine Antwortseite, indem auf alle Tabellennamen zugegriffen wird und jede Seite ausgegeben wird. Nachdem die Seite verarbeitet wurde, wird eine weitere Seite vom Herausgeber angefordert. Diese Methode wird wiederholt aufgerufen, bis alle Seiten abgerufen werden.

Die `onError`-Methode wird ausgelöst, wenn ein Fehler auftritt, während Daten abgerufen werden. Schließlich wird die `onComplete`-Methode aufgerufen, nachdem alle Seiten angefordert wurden.

```
    // A Subscription represents a one-to-one life-cycle of a Subscriber
    subscribing
    // to a Publisher.
    publisher.subscribe(new Subscriber<ListTablesResponse>() {
        // Maintain a reference to the subscription object, which is required to
        request
        // data from the publisher.
        private Subscription subscription;

        @Override
        public void onSubscribe(Subscription s) {
            subscription = s;
            // Request method should be called to demand data. Here we request a
            single
            // page.
            subscription.request(1);
        }

        @Override
        public void onNext(ListTablesResponse response) {
            response.tableNames().forEach(System.out::println);
            // After you process the current page, call the request method to
            signal that
            // you are ready for next page.
            subscription.request(1);
        }

        @Override
        public void onError(Throwable t) {
            // Called when an error has occurred while processing the requests.
        }

        @Override
        public void onComplete() {
            // This indicates all the results are delivered and there are no more
            pages
            // left.
        }
    });
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Verwenden eines **Consumer**

Die `SdkPublisher` Schnittstelle, die `ListTablesPublisher` implementiert, hat eine `subscribe` Methode, die ein verwendet `Consumer` und ein zurückgibt `CompletableFuture<Void>`.

Die `subscribe` Methode von dieser Schnittstelle kann für einfache Anwendungsfälle verwendet werden, in denen ein zu viel Overhead verursachen `org.reactivestreams.Subscriber` könnte. Da der folgende Code jede Seite verbraucht, ruft er die `tableNames` Methode auf jeder Seite auf. Die `tableNames` Methode gibt einen `java.util.List` von DynamoDB-Tabellennamen zurück, die mit der `forEach` Methode verarbeitet werden.

```
// Use a Consumer for simple use cases.
CompletableFuture<Void> future = publisher.subscribe(
    response -> response.tableNames()
        .forEach(System.out::println));
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Iterieren über Tabellennamen

Die folgenden Beispiele zeigen Möglichkeit, um über die in der Antwort zurückgegebenen Objekte anstatt über die Seiten der Antwort zu iterieren. Ähnlich wie das zuvor mit seiner `contents` Methode gezeigte synchrone Amazon S3-Beispiel `ListTablesPublisher` verfügt die asynchrone DynamoDB-Ergebnisklasse über die `tableNames` bequeme Methode, mit der zugrunde liegenden Elementauflistung zu interagieren. Der Rückgabtyp der `tableNames` Methode ist ein [SdkPublisher](#), mit dem Elemente auf allen Seiten angefordert werden können.

## Verwenden eines **Subscriber**

Der folgende Code ruft eine `SdkPublisher` der zugrunde liegenden Sammlung von Tabellennamen ab.

```
// Create a default client with credentials and region loaded from the
// environment.
final DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();

ListTablesRequest listTablesRequest =
ListTablesRequest.builder().limit(3).build();
ListTablesPublisher listTablesPublisher =
asyncClient.listTablesPaginator(listTablesRequest);
SdkPublisher<String> publisher = listTablesPublisher.tableNames();
```

Der folgende Code verwendet eine anonyme Implementierung von `org.reactivestreams.Subscriber`, um die Ergebnisse für jede Seite zu verarbeiten.

Die `onNext` Methode des Abonnenten verarbeitet ein einzelnes Element der Sammlung. In diesem Fall handelt es sich um einen Tabellennamen. Nachdem der Tabellennamen verarbeitet wurde, wird vom Herausgeber ein anderer Tabellennamen angefordert. Diese Methode wird wiederholt aufgerufen, bis alle Tabellennamen abgerufen werden.

```
// Use a Subscriber.
publisher.subscribe(new Subscriber<String>() {
    private Subscription subscription;

    @Override
    public void onSubscribe(Subscription s) {
        subscription = s;
        subscription.request(1);
    }

    @Override
    public void onNext(String tableName) {
        System.out.println(tableName);
        subscription.request(1);
    }

    @Override
    public void onError(Throwable t) {
    }

    @Override
    public void onComplete() {
    }
});
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

### Verwenden eines **Consumer**

Im folgenden Beispiel wird die `subscribe` Methode von `verwendetSdkPublisher`, die ein `Consumer` um jedes Element zu verarbeiten.

```
// Use a Consumer.
CompletableFuture<Void> future = publisher.subscribe(System.out::println);
```

```
future.get();
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Verwenden der Bibliothek eines Drittanbieters

Sie können andere Drittanbieter-Bibliotheken verwenden, statt einen benutzerdefinierten Abonnenten zu implementieren. Dieses Beispiel zeigt die Verwendung von RxJava, aber jede Bibliothek, die die reaktiven Stream-Schnittstellen implementiert, kann verwendet werden. Weitere Informationen zu dieser Bibliothek finden Sie [RxJava auf der Wiki-Seite GitHub](#) auf .

Um die Bibliothek zu verwenden, fügen Sie sie als Abhängigkeit hinzu. Bei Verwendung von Maven zeigt das Beispiel den zu verwendenden POM-Ausschnitt.

### POM-Eintrag

```
<dependency>
  <groupId>io.reactivex.rxjava3</groupId>
  <artifactId>rxjava</artifactId>
  <version>3.1.6</version>
</dependency>
```

### Code

```
DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();
ListTablesPublisher publisher =
asyncClient.listTablesPaginator(ListTablesRequest.builder()
    .build());

// The Flowable class has many helper methods that work with
// an implementation of an org.reactivestreams.Publisher.
List<String> tables = Flowable.fromPublisher(publisher)
    .flatMapIterable(ListTablesResponse::tableNames)
    .toList()
    .blockingGet();
System.out.println(tables);
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

# Abfrage nach Ressourcenstatus im AWS SDK for Java 2.x: Waiters

Mit dem Waiters-Dienstprogramm von AWS SDK for Java 2.x können Sie überprüfen, ob sich AWS Ressourcen in einem bestimmten Status befinden, bevor Sie Vorgänge für diese Ressourcen ausführen.

Ein Waiter ist eine Abstraktion, die verwendet wird, um AWS Ressourcen wie DynamoDB Tabellen oder Amazon S3 Buckets abzufragen, bis ein gewünschter Status erreicht ist (oder bis festgestellt wird, dass die Ressource niemals den gewünschten Status erreicht). Anstatt Logik zu schreiben, um Ihre AWS Ressourcen kontinuierlich abzufragen, was mühsam und fehleranfällig sein kann, können Sie Waiter verwenden, um eine Ressource abzufragen und Ihren Code weiter laufen zu lassen, nachdem die Ressource bereit ist.

## Voraussetzungen

Bevor Sie Waiter in einem Projekt mit der verwenden können AWS SDK for Java, müssen Sie die Schritte unter [Einrichten der AWS SDK for Java 2.x](#) ausführen.

Sie müssen auch Ihre Projektabhängigkeiten konfigurieren (z. B. in Ihrer `-pom.xml` oder `-build.gradle` Datei), um Version `2.15.0` oder höher des zu verwenden AWS SDK for Java.

Beispielsweise:

```
<project>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.15.0</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
</project>
```

## Verwenden von Waitern

Um ein Waiters-Objekt zu instanziiieren, erstellen Sie zunächst einen Service-Client. Legen Sie die `waiter()` Methode des Service-Clients als Wert des Waiter-Objekts fest. Sobald die Waiter-

Instance vorhanden ist, legen Sie ihre Antwortoptionen fest, um den entsprechenden Code auszuführen.

## Synchrone Programmierung

Der folgende Codeausschnitt zeigt, wie Sie warten, bis eine DynamoDB Tabelle vorhanden ist und sich im Status ACTIVE befindet.

```
DynamoDbClient dynamo = DynamoDbClient.create();
DynamoDbWaiter waiter = dynamo.waiter();

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

// print out the matched response with a tableStatus of ACTIVE
waiterResponse.matched().response().ifPresent(System.out::println);
```

## Asynchrone Programmierung

Der folgende Codeausschnitt zeigt, wie Sie warten, bis eine DynamoDB Tabelle nicht mehr existiert.

```
DynamoDbAsyncClient asyncDynamo = DynamoDbAsyncClient.create();
DynamoDbAsyncWaiter asyncWaiter = asyncDynamo.waiter();

CompletableFuture<WaiterResponse<DescribeTableResponse>> waiterResponse =
    asyncWaiter.waitUntilTableNotExists(r -> r.tableName("myTable"));

waiterResponse.whenComplete((r, t) -> {
    if (t == null) {
        // print out the matched ResourceNotFoundException
        r.matched().exception().ifPresent(System.out::println);
    }
}).join();
```

## Konfigurieren von Waitern

Sie können die Konfiguration für einen Waiter anpassen, indem Sie die `overrideConfiguration()` auf seinem Builder verwenden. Bei einigen Operationen können Sie eine benutzerdefinierte Konfiguration anwenden, wenn Sie die Anforderung stellen.



## Konfigurieren eines Waiters

Der folgende Codeausschnitt zeigt, wie Sie die Konfiguration auf einem Waiter überschreiben.

```
// sync
DynamoDbWaiter waiter =
    DynamoDbWaiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(10))
        .client(dynamoDbClient)
        .build();

// async
DynamoDbAsyncWaiter asyncWaiter =
    DynamoDbAsyncWaiter.builder()
        .client(dynamoDbAsyncClient)
        .overrideConfiguration(o -> o.backoffStrategy(
            FixedDelayBackoffStrategy.create(Duration.ofSeconds(2))))
        .scheduledExecutorService(Executors.newScheduledThreadPool(3))
        .build();
```

## Konfiguration für eine bestimmte Anforderung überschreiben

Der folgende Codeausschnitt zeigt, wie die Konfiguration für einen Waiter auf Anforderungsbasis überschrieben wird. Beachten Sie, dass nur einige Operationen anpassbare Konfigurationen haben.

```
waiter.waitUntilTableNotExists(b -> b.tableName("myTable"),
    o -> o.maxAttempts(10));

asyncWaiter.waitUntilTableExists(b -> b.tableName("myTable"),
    o -> o.waitTimeout(Duration.ofMinutes(1)));
```

## Codebeispiele

Ein vollständiges Beispiel für die Verwendung von Waitern mit DynamoDB finden Sie unter [CreateTable.java](#) im AWS Code Examples Repository.

Ein vollständiges Beispiel für die Verwendung von Waitern mit Amazon S3 finden Sie unter [S3BucketOps.java](#) im AWS Code Examples Repository.

## Asynchrone Programmierung verwenden

Die AWS SDK for Java 2.x bietet asynchrone Clients mit nicht blockierender I/O-Unterstützung, die eine hohe Parallelität über einige Threads hinweg implementieren. Die gesamte nicht blockierende E/A ist jedoch nicht garantiert. Der asynchrone Client kann in einigen Fällen blockierende Aufrufe durchführen, z. B. das Abrufen von Anmeldeinformationen, das Signieren von Anforderungen mit [AWS Signature Version 4 \(SigV4\)](#) oder die Endpunkterkennung.

Synchrone Methoden blockieren die Ausführung Ihres Threads, bis der Client eine Antwort vom Service erhält. Asynchrone Methoden kehren sofort zurück. So haben Sie die Gewissheit, dass die Kontrolle an den aufrufenden Thread zurückgegeben wird, ohne auf eine Antwort zu warten.

Da eine asynchrone Methode zurückmeldet, bevor eine Antwort verfügbar ist, benötigen Sie einen Weg, an die Antwort zu gelangen, sobald diese bereitsteht. Die Methoden für den asynchronen Client in 2.x der AWS SDK for Java `CompletableFuture` Rückgabeobjekte, mit denen Sie auf die Antwort zugreifen können, wenn sie bereit ist.

## Nicht-Streaming-Operationen

Bei Nicht-Streaming-Operationen ähneln asynchrone Methodenaufrufe synchronen Methoden. Die asynchronen Methoden im geben jedoch ein [CompletableFuture](#) Objekt AWS SDK for Java zurück, das die Ergebnisse der asynchronen Operation in der Zukunft enthält.

Rufen Sie die `-CompletableFuture.whenComplete()` Methode mit einer `-Aktion` auf, um den Vorgang abzuschließen, wenn das Ergebnis verfügbar ist. `CompletableFuture` implementiert die `-FutureSchnittstelle`, sodass Sie das Antwortobjekt auch abrufen können, indem Sie die `-get()` Methode aufrufen.

Im Folgenden finden Sie ein Beispiel für eine asynchrone `-Operation`, die eine `- Amazon DynamoDB Funktion` aufruft, um eine Liste von Tabellen abzurufen, die ein `empfängtCompletableFuture`, das ein [ListTablesResponse](#) Objekt enthalten kann. Die im Aufruf von `whenComplete()` definierte Aktion wird nur ausgeführt, wenn der asynchrone Aufruf abgeschlossen ist.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
```

```
import java.util.List;
import java.util.concurrent.CompletableFuture;
```

## Code

```
public class DynamoDBAsyncListTables {

    public static void main(String[] args) throws InterruptedException {

        // Create the DynamoDbAsyncClient object
        Region region = Region.US_EAST_1;
        DynamoDbAsyncClient client = DynamoDbAsyncClient.builder()
            .region(region)
            .build();

        listTables(client);
    }

    public static void listTables(DynamoDbAsyncClient client) {

        CompletableFuture<ListTablesResponse> response =
client.listTables(ListTablesRequest.builder()
            .build());

        // Map the response to another CompletableFuture containing just the table
names
        CompletableFuture<List<String>> tableNames =
response.thenApply(ListTablesResponse::tableNames);

        // When future is complete (either successfully or in error) handle the
response
        tableNames.whenComplete((tables, err) -> {
            try {
                if (tables != null) {
                    tables.forEach(System.out::println);
                } else {
                    // Handle error
                    err.printStackTrace();
                }
            } finally {
                // Lets the application shut down. Only close the client when you are
completely done with it.
                client.close();
            }
        });
    }
}
```

```
        }
    });
    tableNames.join();
}
}
```

Im folgenden Codebeispiel wird gezeigt, wie ein Element mithilfe des asynchronen Clients aus einer Tabelle abgerufen wird. Rufen Sie die `-getItem`-Methode des `DynamoDbAsyncClient` auf und übergeben Sie ihm ein [GetItemRequest](#) Objekt mit dem Tabellennamen und dem Primärschlüsselwert des gewünschten Elements. Dies ist in der Regel, wie Sie Daten übergeben, die für den Vorgang erforderlich sind. Beachten Sie in diesem Beispiel, dass ein Zeichenfolgenwert übergeben wird.

## Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

## Code

```
public static void getItem(DynamoDbAsyncClient client, String tableName, String
key, String keyVal) {

    HashMap<String, AttributeValue> keyToGet =
        new HashMap<String, AttributeValue>();

    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal).build());

    try {

        // Create a GetItemRequest instance
        GetItemRequest request = GetItemRequest.builder()
            .key(keyToGet)
            .tableName(tableName)
```

```
        .build();

        // Invoke the DynamoDbAsyncClient object's getItem
        java.util.Collection<AttributeValue> returnedItem =
client.getItem(request).join().item().values();

        // Convert Set to Map
        Map<String, AttributeValue> map =
returnedItem.stream().collect(Collectors.toMap(AttributeValue::s, s->s));
        Set<String> keys = map.keySet();
        for (String sinKey : keys) {
            System.out.format("%s: %s\n", sinKey, map.get(sinKey).toString());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Streaming-Operationen

Für Streaming-Operationen müssen Sie einen angeben, [AsyncRequestBody](#) um den Inhalt inkrementell bereitzustellen, oder einen , [AsyncResponseTransformer](#) um die Antwort zu empfangen und zu verarbeiten.

Im folgenden Beispiel wird eine Datei Amazon S3 asynchron unter Verwendung der `-PutObjectOperation` in hochgeladen.

### Importe

```
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import java.nio.file.Paths;
import java.util.concurrent.CompletableFuture;
```

### Code

```
/**
```

```
* To run this AWS code example, ensure that you have setup your development
environment, including your AWS credentials.
*
* For information, see this documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
```

```
public class S3AsyncOps {

    public static void main(String[] args) {

        final String USAGE = "\n" +
            "Usage:\n" +
            "    S3AsyncOps <bucketName> <key> <path>\n\n" +
            "Where:\n" +
            "    bucketName - the name of the Amazon S3 bucket (for example,
bucket1). \n\n" +
            "    key - the name of the object (for example, book.pdf). \n" +
            "    path - the local path to the file (for example, C:/AWS/book.pdf).
\n" ;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String key = args[1];
        String path = args[2];

        Region region = Region.US_WEST_2;
        S3AsyncClient client = S3AsyncClient.builder()
            .region(region)
            .build();

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        // Put the object into the bucket
        CompletableFuture<PutObjectResponse> future = client.putObject(objectRequest,
            AsyncRequestBody.fromFile(Paths.get(path)))
```

```
    );
    future.whenComplete((resp, err) -> {
        try {
            if (resp != null) {
                System.out.println("Object uploaded. Details: " + resp);
            } else {
                // Handle error
                err.printStackTrace();
            }
        } finally {
            // Only close the client when you are completely done with it
            client.close();
        }
    });

    future.join();
}
}
```

Im folgenden Beispiel wird eine -Datei Amazon S3 asynchron von mithilfe der -GetObjectOperation abgerufen.

## Importe

```
import software.amazon.awssdk.core.async.AsyncResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.nio.file.Paths;
import java.util.concurrent.CompletableFuture;
```

## Code

```
/**
 * To run this AWS code example, ensure that you have setup your development
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class S3AsyncStreamOps {

    public static void main(String[] args) {

        final String USAGE = "\n" +
            "Usage:\n" +
            "  S3AsyncStreamOps <bucketName> <objectKey> <path>\n\n" +
            "Where:\n" +
            "  bucketName - the name of the Amazon S3 bucket (for example,
bucket1). \n\n" +
            "  objectKey - the name of the object (for example, book.pdf). \n" +
            "  path - the local path to the file (for example, C:/AWS/book.pdf).
\n" ;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String path = args[2];

        Region region = Region.US_WEST_2;
        S3AsyncClient client = S3AsyncClient.builder()
            .region(region)
            .build();

        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        CompletableFuture<GetObjectResponse> futureGet =
client.getObject(objectRequest,
    AsyncResponseTransformerToFile(Paths.get(path)));

        futureGet.whenComplete((resp, err) -> {
            try {
                if (resp != null) {
                    System.out.println("Object downloaded. Details: "+resp);
                } else {
                    err.printStackTrace();
                }
            }
        })
    }
```



```
        } finally {
            // Only close the client when you are completely done with it
            client.close();
        }
    });
    futureGet.join();
}
}
```

## Erweiterte Operationen

Das AWS SDK for Java 2.x verwendet [Netty](#), ein asynchrones ereignisgesteuertes Netzwerkanwendungs-Framework, um E/A-Threads zu verarbeiten. Die AWS SDK for Java 2.x erstellt eine `ExecutorService` hinter Netty, um die von der HTTP-Client-Anforderung bis zum Netty-Client zurückgegebenen futures abzuschließen. Diese Abstraktion reduziert das Risiko, dass eine Anwendung den asynchronen Prozess unterbricht, wenn Entwickler Threads anhalten oder in den Ruhezustand versetzen. Standardmäßig erstellt jeder asynchrone Client basierend auf der Anzahl der Prozessoren einen Threadpool und verwaltet die Aufgaben in einer Warteschlange innerhalb der `ExecutorService`.

Fortgeschrittene Benutzer können ihre Thread-Pool-Größe beim Erstellen eines asynchronen Clients mit der folgenden Option beim Erstellen angeben.

### Code

```
S3AsyncClient clientThread = S3AsyncClient.builder()
    .asyncConfiguration(
        b -> b.advancedOption(SdkAdvancedAsyncClientOption
            .FUTURE_COMPLETION_EXECUTOR,
            Executors.newFixedThreadPool(10)
        )
    )
    .build();
```

Um die Leistung zu optimieren, können Sie Ihren eigenen Thread-Pool-Executor verwalten und ihn bei der Konfiguration Ihres Clients einschließen.

```
ThreadPoolExecutor executor = new ThreadPoolExecutor(50, 50,
    10, TimeUnit.SECONDS,
    new LinkedBlockingQueue<>(<custom_value>),
    new ThreadFactoryBuilder()
```

```
        .threadNamePrefix("sdk-async-response").build());

// Allow idle core threads to time out
executor.allowCoreThreadTimeOut(true);

S3AsyncClient clientThread = S3AsyncClient.builder()
    .asyncConfiguration(
        b -> b.advancedOption(SdkAdvancedAsyncClientOption
            .FUTURE_COMPLETION_EXECUTOR,
            executor
        )
    )
    .build();
```

## Arbeiten mit HTTP/2 im AWS SDK for Java

HTTP/2 ist eine Hauptversion des HTTP-Protokolls. Diese neue Version verfügt über mehrere Erweiterungen, um die Leistung zu verbessern:

- Binäre Datencodierung ermöglicht eine effizientere Datenübertragung.
- Header-Komprimierung reduziert die vom Client heruntergeladenen Overhead-Bytes, sodass die Inhalte früher zum Client gelangen. Dies ist besonders bei mobilen Clients nützlich, bei denen die Bandbreite bereits eingeschränkt ist.
- Bidirektionale asynchrone Kommunikation (Multiplexing) ermöglicht es AWS, mehrere Anfragen und Antwortnachrichten zwischen dem Client und über eine einzige Verbindung gleichzeitig zu übertragen, anstatt über mehrere Verbindungen, was die Leistung verbessert.

Entwickler, die auf die neuesten SDKs aktualisieren, verwenden automatisch HTTP/2, wenn es von dem Service, mit dem sie arbeiten, unterstützt wird. Neue Programmierschnittstellen nutzen nahtlos die Vorteile von HTTP/2 Funktionen und bieten neue Möglichkeiten zur Erstellung von Anwendungen.

Die AWS SDK for Java 2.x bietet neue APIs für das Ereignis-Streaming, die das HTTP/2-Protokoll implementieren. Beispiele für die Verwendung dieser neuen APIs finden Sie unter [Arbeiten mit Kinesis](#).

## Verwenden von SDK-Metriken aus der AWS SDK for Java

Mit dem AWS SDK for Java 2.x können Sie Metriken über die Service-Clients in Ihrer Anwendung erfassen, die Ausgabe in analysieren Amazon CloudWatch und dann darauf reagieren.

Standardmäßig ist die Erfassung von Metriken im SDK deaktiviert. Dieses Thema hilft Ihnen, es zu aktivieren und zu konfigurieren.

## Voraussetzungen

Bevor Sie Metriken aktivieren und verwenden können, müssen Sie die folgenden Schritte ausführen:

- Führen Sie die Schritte unter [Aufstellen](#) aus.
- Konfigurieren Sie Ihre Projektabhängigkeiten (z. B. in Ihrer `-pom.xml` oder `-build.gradle` Datei), um Version `2.14.0` oder höher des zu verwenden AWS SDK for Java.

Um die Veröffentlichung von Metriken in zu ermöglichen CloudWatch, fügen Sie auch die `artifactId` `cloudwatch-metric-publisher` mit der Versionsnummer `2.14.0` oder höher in die Abhängigkeiten Ihres Projekts ein.

Beispielsweise:

```
<project>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.14.0</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>cloudwatch-metric-publisher</artifactId>
      <version>2.14.0</version>
    </dependency>
  </dependencies>
</project>
```

- Aktivieren Sie `cloudwatch:PutMetricData` Berechtigungen für die IAM-Identität, damit das SDK for Java Metriken schreiben kann.

## So aktivieren Sie die Erfassung von Metriken

Sie können Metriken in Ihrer Anwendung für einen Service-Client oder für einzelne Anforderungen aktivieren.

### Aktivieren von Metriken für eine bestimmte Anforderung

Der folgende Codeausschnitt zeigt, wie Sie den CloudWatch Metrikherausgeber für eine Anforderung an aktivieren Amazon DynamoDB. Es verwendet die Standardkonfiguration des Metrikherausgebers.

```
MetricPublisher metricsPub = CloudWatchMetricPublisher.create();
DynamoDbClient ddb = DynamoDbClient.create();
ddb.listTables(ListTablesRequest.builder()
    .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
    .build());
```

### Aktivieren von Metriken für einen bestimmten Service-Client

Der folgende Codeausschnitt zeigt, wie Sie einen CloudWatch Metrikherausgeber mit Standardeinstellungen für einen Service-Client aktivieren.

```
MetricPublisher metricsPub = CloudWatchMetricPublisher.create();

DynamoDbClient ddb = DynamoDbClient.builder()
    .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
    .build();
```

Der folgende Codeausschnitt zeigt, wie eine benutzerdefinierte Konfiguration für den Metrikherausgeber für einen bestimmten Service-Client verwendet wird. Zu den Anpassungen gehören das Laden eines bestimmten Anmeldeinformationsprofils, die Angabe einer anderen Region als der Service-Client und die Anpassung der Häufigkeit, mit der der Publisher Metriken an sendet CloudWatch.

```
MetricPublisher metricsPub = CloudWatchMetricPublisher.builder()
    .cloudWatchClient(CloudWatchAsyncClient.builder()
        .region(Region.US_WEST_2)

        .credentialsProvider(ProfileCredentialsProvider.create("cloudwatch"))
        .build())
    .uploadFrequency(Duration.ofMinutes(5))
```

```
.build();

DynamoDbClient ddb = DynamoDbClient.builder()
    .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
    .build();
```

## Wann sind Metriken verfügbar?

Metriken sind im Allgemeinen innerhalb von 5 bis 10 Minuten nach der Ausgabe durch das SDK for Java verfügbar. Um genaue - und - up-to-date Metriken zu erhalten, überprüfen Sie Cloudwatch mindestens 10 Minuten, nachdem Sie die Metriken aus Ihren Java-Anwendungen ausgegeben haben.

## Welche Informationen werden erfasst?

Die Erfassung von Metriken umfasst Folgendes:

- Anzahl der API-Anforderungen, einschließlich ob sie erfolgreich sind oder fehlschlagen
- Informationen zu den AWS Services, die Sie in Ihren API-Anforderungen aufrufen, einschließlich zurückgegebener Ausnahmen
- Die Dauer für verschiedene Operationen wie z. B. Verteilen, Signieren und HTTP-Anfragen
- HTTP-Client-Metriken, z. B. die Anzahl der offenen Verbindungen, die Anzahl der ausstehenden Anforderungen und der Name des verwendeten HTTP-Clients

### Note

Die verfügbaren Metriken variieren je nach HTTP-Client.

Eine vollständige Liste finden Sie unter [Service-Client-Metriken](#).

## Wie kann ich diese Informationen verwenden?

Sie können die Metriken verwenden, die das SDK sammelt, um die Service-Clients in Ihrer Anwendung zu überwachen. Sie können sich allgemeine Nutzungstrends ansehen, Anomalien identifizieren, die zurückgegebenen Service-Client-Ausnahmen überprüfen oder ein bestimmtes Problem nachvollziehen. Mit können Sie auch Alarme erstellen Amazon CloudWatch, um Sie zu benachrichtigen, sobald Ihre Anwendung eine von Ihnen definierte Bedingung erreicht.

Weitere Informationen finden Sie unter [Verwenden von Amazon CloudWatch Metriken](#) und [Verwenden von Amazon CloudWatch Alarmen](#) im [Amazon CloudWatch -Benutzerhandbuch](#).

## Service-Client-Metriken

Mit der können Sie Metriken von den Service- AWS SDK for Java 2.x Clients in Ihrer Anwendung erfassen und diese Metriken dann in [Amazon CloudWatch](#) veröffentlichen (ausgeben).

In diesen Tabellen sind die Metriken aufgeführt, die Sie erfassen können, sowie alle HTTP-Client-Nutzungsanforderungen.

Weitere Informationen zum Aktivieren und Konfigurieren von Metriken für das SDK finden Sie unter [Aktivieren von SDK-Metriken](#).

Die in den Tabellen verwendeten Begriffe bedeuten:

- Apache: Der Apache-basierte HTTP-Client ([ApacheHttpClient](#))
- Netty: der Netty-basierte HTTP-Client ([NettyNioAsyncHttpClient](#))
- CRT: der AWS CRT-basierte HTTP-Client ([AwsCrtAsyncHttpClient](#))
- Beliebig: Die Erfassung von Metrikdaten hängt nicht vom HTTP-Client ab; dazu gehört auch die Verwendung des URLConnection-basierten HTTP-Clients ([URLConnectionHttpClient](#))

### Bei jeder Anforderung erfasste Metriken

Metrikname	Beschreibung	Typ	HTTP-Client erforderlich
ApiCallDuration	Die Gesamtzeit, die benötigt wird, um eine Anfrage abzuschließen (einschließlich aller Wiederholungen)	Dauer	Any
ApiCallSuccessful	True, wenn der API-Aufruf erfolgreich war, false, wenn nicht	Boolesch	Any

Metrikname	Beschreibung	Typ	HTTP-Client erforderlich
CredentialsFetchDuration	Die Zeit, die zum Abrufen der AWS Signaturanmeldeinformationen für die Anforderung benötigt wird	Dauer	Any
MarshallingDuration	Die Zeit, die benötigt wird, um die Anfrage zu bearbeiten	Dauer	Any
OperationName	Der Name der AWS API, an die die Anforderung gestellt wird	String	Any
RetryCount	Gibt an, wie oft das SDK den API-Aufruf wiederholt hat	Ganzzahl	Any
ServiceId	Service-ID der AWS-Service, für die die API-Anforderung gestellt wird	String	Any
TokenFetchDuration	Die Zeit, die zum Abrufen der Anmeldeinformationen für die Token-Signatur für die Anforderung benötigt wird	Dauer	Any

## Für jeden Anforderungsversuch gesammelte Metriken

Jeder API-Aufruf erfordert möglicherweise mehrere Versuche, bevor eine Antwort empfangen wird. Diese Metriken werden für jeden Versuch erfasst.

Metrikname	Beschreibung	Typ	HTTP-Client erforderlich
AvailableConcurrency	Die Anzahl der verbleibenden gleichzeitigen Anforderungen, die vom HTTP-Client unterstützt werden können, ohne eine weitere Verbindung herstellen zu müssen	Ganzzahl	Apache, Netty, CRT
AwsExtendedRequestId	Die erweiterte Anforderungs-ID der Serviceanforderung	String	Any
AwsRequestId	Die Anforderungs-ID der Serviceanforderung	String	Any
BackoffDelayDuration	Die Zeitspanne, die das SDK vor diesem API-Aufrufversuch gewartet hat	Dauer	Any
ConcurrencyAcquireDuration	Die Zeit, die benötigt wird, um einen Kanal aus dem Verbindungspool zu erwerben	Dauer	Apache, Netty, CRT



Metrikname	Beschreibung	Typ	HTTP-Client erforderlich
HttpClientName	Der Name des HTTP, das für die Anforderung verwendet wird	String	Apache, Netty, CRT
HttpStatusCode	Der Statuscode, der mit der HTTP-Antwort zurückgegeben wird	Ganzzahl	Any
LeasedConcurrency	Die Anzahl der Anforderungen, die derzeit vom HTTP-Client ausgeführt werden	Ganzzahl	Apache, Netty, CRT
LocalStreamWindowSize	Die lokale HTTP/2-Fenstergröße in Byte für den Stream, für den diese Anforderung ausgeführt wurde	Ganzzahl	Netty
MarshallingDuration	Die Zeit, die benötigt wird, um eine -SDK-Anfrage auf eine HTTP-Anfrage zu beschränken	Dauer	Any
MaxConcurrency	Die maximale Anzahl gleichzeitiger Anforderungen, die vom HTTP-Client unterstützt werden	Ganzzahl	Apache, Netty, CRT

Metrikname	Beschreibung	Typ	HTTP-Client erforderlich
PendingConcurrencyAcquires	Die Anzahl der blockierten Anforderungen, die darauf warten, dass eine andere TCP-Verbindung oder ein neuer Stream aus dem Verbindungspool verfügbar ist	Ganzzahl	Apache, Netty, CRT
RemoteStreamWindowSize	Die Remote-HTTP/2-Fenstergröße in Byte für den Stream, für den diese Anforderung ausgeführt wurde	Ganzzahl	Netty
ServiceCallDuration	Die Zeit, die benötigt wird, um eine Verbindung zum Service herzustellen, die Anforderung zu senden und den HTTP-Statuscode und den Header aus der Antwort zu erhalten	Dauer	Any
SigningDuration	Die Zeit, die zum Signieren der HTTP-Anforderung benötigt wird	Dauer	Any

Metrikname	Beschreibung	Typ	HTTP-Client erforderlich
UnmarshallingDuration	Die Zeit, die benötigt wird, um eine HTTP-Antwort auf eine SDK-Antwort aufzuheben	Dauer	Any

# Arbeiten mit AWS-Services unter Verwendung der AWS SDK for Java 2.x

Dieser Abschnitt enthält kurze Tutorials und Anleitungen zur Arbeit mit Select AWS-Services. Eine vollständige Reihe von Beispielen finden Sie im [Abschnitt Codebeispiele](#).

## Themen

- [Arbeiten mit CloudWatch](#)
- [AWS -Datenbankservices und AWS SDK for Java 2.x](#)
- [Arbeiten mit DynamoDB](#)
- [Arbeiten mit Amazon EC2](#)
- [Arbeiten mit IAM](#)
- [Arbeiten mit Kinesis](#)
- [Aufrufen, Auflisten und Löschen AWS Lambda von Funktionen](#)
- [Arbeiten mit Amazon S3](#)
- [Arbeiten mit Amazon Simple Notification Service](#)
- [Arbeiten mit Amazon Simple Queue Service](#)
- [Arbeiten mit Amazon Transcribe](#)

## Arbeiten mit CloudWatch

Dieser Abschnitt enthält Beispiele für die Programmierung CloudWatch von [Amazon](#) mithilfe von AWS SDK for Java 2.x.

Amazon CloudWatch überwacht Ihre Amazon Web Services (AWS)-Ressourcen und die Anwendungen, die Sie in AWS ausführen, in Echtzeit. Sie können CloudWatch verwenden, um Metriken zu erfassen und nachzuverfolgen. Es handelt sich dabei um Variablen, die Sie für Ihre Ressourcen und Anwendungen messen können. Abhängig von den von Ihnen definierten Regeln senden CloudWatch-Alarme Benachrichtigungen oder nehmen automatisch Änderungen an den von Ihnen überwachten Ressourcen vor.

Die folgenden Beispiele enthalten nur den Code, der zur Demonstration jeder Technik nötig ist. Der [vollständige Beispielcode ist verfügbar unter GitHub](#). Von dort aus können Sie eine einzelne

Quelldatei herunterladen oder das Repository klonen, um alle Beispiele lokal zu erstellen und auszuführen.

## Themen

- [Abrufen von Metriken von CloudWatch](#)
- [Veröffentlichen von benutzerdefinierten Metrikdaten in CloudWatch](#)
- [Arbeiten mit CloudWatch Alarmen](#)
- [Verwenden von Amazon CloudWatch Events](#)

## Abrufen von Metriken von CloudWatch

### Auflisten von Metriken

Um CloudWatch Metriken aufzulisten, erstellen Sie ein [ListMetricsRequest](#) und rufen Sie die `listMetrics` Methode `CloudWatchClient` auf. Sie können `ListMetricsRequest` zum Filtern der zurückgegebenen Metriken nach Namespace, Metrikname oder Dimensionen verwenden.

#### Note

Eine Liste der Metriken und Dimensionen, die von `-AWSServices` veröffentlicht werden, finden Sie in der [Amazon CloudWatch Referenz zu Metriken und Dimensionen](#) im Amazon CloudWatch-Benutzerhandbuch.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;
```

### Code

```
public static void listMets( CloudWatchClient cw, String namespace) {

    boolean done = false;
```

```
String nextToken = null;

try {
    while(!done) {

        ListMetricsResponse response;

        if (nextToken == null) {
            ListMetricsRequest request = ListMetricsRequest.builder()
                .namespace(namespace)
                .build();

            response = cw.listMetrics(request);
        } else {
            ListMetricsRequest request = ListMetricsRequest.builder()
                .namespace(namespace)
                .nextToken(nextToken)
                .build();

            response = cw.listMetrics(request);
        }

        for (Metric metric : response.metrics()) {
            System.out.printf(
                "Retrieved metric %s", metric.metricName());
            System.out.println();
        }

        if(response.nextToken() == null) {
            done = true;
        } else {
            nextToken = response.nextToken();
        }
    }

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

Die Metriken werden in einem [ListMetricsResponse](#) durch Aufrufen seiner `getMetrics` Methode zurückgegeben.

Eventuell werden die Ergebnisse seitenweise zurückgegeben. Um den nächsten Stapel Ergebnisse abzurufen, rufen Sie `nextToken` für das Antwortobjekt auf und verwenden den Tokenwert, um ein neues Anforderungsobjekt zu erstellen. Anschließend rufen Sie die `listMetrics`-Methode erneut für die neue Anforderung auf.

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Weitere Informationen

- [ListMetrics](#) in der Amazon CloudWatch API-Referenz zu

## Veröffentlichen von benutzerdefinierten Metrikdaten in CloudWatch

Eine Reihe von AWS Services veröffentlicht [ihre eigenen Metriken](#) in Namespaces, die mit „`AWS`“ beginnen, können Sie auch benutzerdefinierte Metrikdaten mit Ihrem eigenen Namespace veröffentlichen (solange sie nicht mit „`AWS`“ beginnen).

## Veröffentlichen benutzerdefinierter Metrikdaten

Um Ihre eigenen Metrikdaten zu veröffentlichen, rufen Sie die `putMetricData`-Methode `CloudWatchClient` des mit einem auf [PutMetricDataRequest](#). muss den benutzerdefinierten Namespace `PutMetricDataRequest` enthalten, der für die Daten verwendet werden soll, sowie Informationen über den Datenpunkt selbst in einem [MetricDatum](#) Objekt.

### Note

Sie können keinen Namespace angeben, der mit „`AWS`“ beginnt. Namespaces, die mit `AWS` " beginnen, sind für die Verwendung durch Amazon Web Services-Produkte reserviert.

## Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.MetricDatum;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
```

```
import java.time.Instant;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
```

## Code

```
public static void putMetData(CloudWatchClient cw, Double dataPoint ) {

    try {
        Dimension dimension = Dimension.builder()
            .name("UNIQUE_PAGES")
            .value("URLS")
            .build();

        // Set an Instant object
        String time =
            ZonedDateTime.now( ZoneOffset.UTC ).format( DateTimeFormatter.ISO_INSTANT );
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName("PAGES_VISITED")
            .unit(StandardUnit.NONE)
            .value(dataPoint)
            .timestamp(instant)
            .dimensions(dimension).build();

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace("SITE/TRAFFIC")
            .metricData(datum).build();

        cw.putMetricData(request);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.printf("Successfully put data point %f", dataPoint);
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.



## Weitere Informationen

- [Verwenden Sie Amazon CloudWatch Metriken](#) im Amazon CloudWatch -Benutzerhandbuch.
- [AWS Namespaces](#) im Amazon CloudWatch -Benutzerhandbuch.
- [PutMetricData](#) in der Amazon CloudWatch API-Referenz zu .

## Arbeiten mit CloudWatch Alarmen

### Alarm erstellen

Um einen Alarm basierend auf einer CloudWatch Metrik zu erstellen, rufen Sie die Methode `CloudWatchClient.putMetricAlarm` mit einem auf, der mit den Alarmbedingungen [PutMetricAlarmRequest](#) gefüllt ist.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricAlarmRequest;
import software.amazon.awssdk.services.cloudwatch.model.ComparisonOperator;
import software.amazon.awssdk.services.cloudwatch.model.Statistic;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
```

### Code

```
public static void putMetricAlarm(CloudWatchClient cw, String alarmName, String
instanceId) {

    try {
        Dimension dimension = Dimension.builder()
            .name("InstanceId")
            .value(instanceId).build();

        PutMetricAlarmRequest request = PutMetricAlarmRequest.builder()
            .alarmName(alarmName)
            .comparisonOperator(
                ComparisonOperator.GREATER_THAN_THRESHOLD)
            .evaluationPeriods(1)
            .metricName("CPUUtilization")
```

```

        .namespace("AWS/EC2")
        .period(60)
        .statistic(Statistic.AVERAGE)
        .threshold(70.0)
        .actionsEnabled(false)
        .alarmDescription(
            "Alarm when server CPU utilization exceeds 70%")
        .unit(StandardUnit.SECONDS)
        .dimensions(dimension)
        .build();

    cw.putMetricAlarm(request);
    System.out.printf(
        "Successfully created alarm with name %s", alarmName);

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Auflisten von Alarmen

Um die von Ihnen erstellten CloudWatch Alarme aufzulisten, rufen Sie die Methode `CloudWatchClient` `describeAlarms` mit einem auf [DescribeAlarmsRequest](#), mit dem Sie Optionen für das Ergebnis festlegen können.

### Importe

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsResponse;
import software.amazon.awssdk.services.cloudwatch.model.MetricAlarm;

```

### Code

```

public static void desCWAlarms( CloudWatchClient cw) {

    try {

```

```
boolean done = false;
String newToken = null;

while(!done) {
    DescribeAlarmsResponse response;

    if (newToken == null) {
        DescribeAlarmsRequest request =
DescribeAlarmsRequest.builder().build();
        response = cw.describeAlarms(request);
    } else {
        DescribeAlarmsRequest request = DescribeAlarmsRequest.builder()
            .nextToken(newToken)
            .build();
        response = cw.describeAlarms(request);
    }

    for(MetricAlarm alarm : response.metricAlarms()) {
        System.out.printf("\n Retrieved alarm %s", alarm.alarmName());
    }

    if(response.nextToken() == null) {
        done = true;
    } else {
        newToken = response.nextToken();
    }
}

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.printf("Done");
}
```

Die Liste der Alarme kann abgerufen werden, indem `MetricAlarms` Sie für die aufrufen [DescribeAlarmsResponse](#), die von zurückgegeben wird `describeAlarms`.

Eventuell werden die Ergebnisse seitenweise zurückgegeben. Um den nächsten Stapel Ergebnisse abzurufen, rufen Sie `nextToken` für das Antwortobjekt auf und verwenden den Tokenwert, um ein neues Anforderungsobjekt zu erstellen. Anschließend rufen Sie die `describeAlarms`-Methode erneut für die neue Anforderung auf.

**Note**

Sie können Alarme für eine bestimmte Metrik auch abrufen, indem Sie die Methode `describeAlarmsForMetric` des `CloudWatchClient` verwenden. Sie lässt sich ähnlich wie `describeAlarms` nutzen.

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Löschen von Alarmen

Um CloudWatch Alarme zu löschen, rufen Sie die Methode `deleteAlarms` des `CloudWatchClient` mit einem auf, der einen oder mehrere Namen von Alarmen [DeleteAlarmsRequest](#) enthält, die Sie löschen möchten.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;
```

### Code

```
public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
    try {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.deleteAlarms(request);
        System.out.printf("Successfully deleted alarm %s", alarmName);
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Weitere Informationen

- [Verwenden von Amazon CloudWatch Alarmen](#) im Amazon CloudWatch -Benutzerhandbuch
- [PutMetricAlarm](#) in der Amazon CloudWatch API-Referenz zu
- [DescribeAlarms](#) in der Amazon CloudWatch API-Referenz zu
- [DeleteAlarms](#) in der Amazon CloudWatch API-Referenz zu

## Verwenden von Amazon CloudWatch Events

CloudWatch Ereignisse bieten einen Stream von Systemereignissen in nahezu Echtzeit, der Änderungen an AWS Ressourcen an Amazon EC2 Instances, Lambda Funktionen, Kinesis Streams, Amazon ECS Aufgaben, Step Functions Zustandsmaschinen, Amazon SNS Themen, Amazon SQS Warteschlangen oder integrierten Zielen beschreibt. Sie können Ereignisse zuordnen und sie zu einer oder mehreren Zielfunktionen oder Streams umleiten, indem Sie einfache Regeln nutzen.

Amazon EventBridge ist die [Entwicklung](#) von CloudWatch Events. Beide Services verwenden dieselbe API, sodass Sie weiterhin den vom SDK bereitgestellten [CloudWatch Events-Client](#) verwenden oder zum [EventBridge Client](#) des SDK für Java für CloudWatch die Events-Funktionalität migrieren können. Die Dokumentation zum CloudWatch Events-Benutzerhandbuch und die [API-Referenz](#) sind jetzt über die EventBridge Dokumentationsseiten verfügbar. <https://docs.aws.amazon.com/eventbridge/latest/userguide/index.html>

## Hinzufügen von Ereignissen

Um benutzerdefinierte CloudWatch Ereignisse hinzuzufügen, rufen Sie die `-CloudWatchEventsClient` 's `putEvents` Methode mit einem [PutEventsRequest](#) Objekt auf, das ein oder mehrere [PutEventsRequestEntry](#) Objekte enthält, die Details zu den einzelnen Ereignissen enthalten. Sie können mehrere Parameter für den Eintrag angeben, wie z. B. die Quelle und den Typ des Ereignisses, mit dem Ereignis verknüpfte Ressourcen usw.

### Note

Sie können maximal 10 Ereignisse pro Aufruf von `putEvents` angeben.

## Importe

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
```

```
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequestEntry;
```

## Code

```
public static void putCWEvents(CloudWatchEventsClient cwe, String resourceArn ) {

    try {

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

        PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
            .detail(EVENT_DETAILS)
            .detailType("sampleSubmitted")
            .resources(resourceArn)
            .source("aws-sdk-java-cloudwatch-example")
            .build();

        PutEventsRequest request = PutEventsRequest.builder()
            .entries(requestEntry)
            .build();

        cwe.putEvents(request);
        System.out.println("Successfully put CloudWatch event");

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Hinzufügen von Regeln

Um eine Regel zu erstellen oder zu aktualisieren, rufen Sie die `CloudWatchEventsClient`'s `putRule` Methode mit einem [PutRuleRequest](#) mit dem Namen der Regel und optionalen Parametern wie dem [Ereignismuster](#), IAM role, die der Regel zugeordnet werden soll, und einem [Planungsausdruck](#) auf, der beschreibt, wie oft die Regel ausgeführt wird.

## Importe

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.RuleState;
```

## Code

```
public static void putCWRule(CloudWatchEventsClient cwe, String ruleName, String
roleArn) {

    try {
        PutRuleRequest request = PutRuleRequest.builder()
            .name(ruleName)
            .roleArn(roleArn)
            .scheduleExpression("rate(5 minutes)")
            .state(RuleState.ENABLED)
            .build();

        PutRuleResponse response = cwe.putRule(request);
        System.out.printf(
            "Successfully created CloudWatch events rule %s with arn %s",
            ruleArn, response.ruleArn());
    } catch (
        CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Hinzufügen von Zielen

Ziele sind die Ressourcen, die beim Auslösen einer Regel aufgerufen werden. Beispiele für Ziele sind Amazon EC2 Instances, Lambda Funktionen, Kinesis Streams, Amazon ECS Aufgaben, Step Functions Zustandsautomaten und integrierte Ziele.

Um einer Regel ein Ziel hinzuzufügen, rufen Sie die `CloudWatchEventsClient`'s `putTargets` Methode mit einem auf, der die zu aktualisierende Regel [PutTargetsRequest](#) enthält, und eine Liste der Ziele, die der Regel hinzugefügt werden sollen.

## Importe

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.Target;
```

## Code

```
public static void putCWTargets(CloudWatchEventsClient cwe, String ruleName, String
functionArn, String targetId ) {

    try {
        Target target = Target.builder()
            .arn(functionArn)
            .id(targetId)
            .build();

        PutTargetsRequest request = PutTargetsRequest.builder()
            .targets(target)
            .rule(ruleName)
            .build();

        PutTargetsResponse response = cwe.putTargets(request);
        System.out.printf(
            "Successfully created CloudWatch events target for rule %s",
            ruleName);
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Weitere Informationen

- [Hinzufügen von Ereignissen mit PutEvents](#) im Amazon EventBridge -Benutzerhandbuch
- [Planen von Ausdrücken für Regeln](#) im Amazon EventBridge -Benutzerhandbuch
- [Ereignistypen für CloudWatch Events](#) im Amazon EventBridge -Benutzerhandbuch



- [Ereignismuster](#) im Amazon EventBridge -Benutzerhandbuch
- [PutEvents](#) in der Amazon EventBridge -API-Referenz zu
- [PutTargets](#) in der Amazon EventBridge -API-Referenz zu
- [PutRule](#) in der Amazon EventBridge -API-Referenz zu

## AWS -Datenbankservices und AWS SDK for Java 2.x

AWS bietet verschiedene Datenbanktypen: relational, Schlüsselwert, In-Memory, Dokument und [mehrere andere](#) . Die Unterstützung des SDK für Java 2.x variiert je nach Art des Datenbankservices in AWS.

Einige Datenbankservices, z. B. der [Amazon-DynamoDB](#)-Service, verfügen über Webservice-APIs zur Verwaltung der AWS Ressource (Datenbank), sowie über Webservice-APIs zur Interaktion mit den Daten. Im SDK für Java 2.x verfügen diese Arten von Services über dedizierte Service-Clients, z. B. [DynamoDBClient](#).

Andere Datenbankservices verfügen über Webservice-APIs, die mit der Ressource interagieren, z. B. die [Amazon DocumentDB](#)-API (für die Cluster-, Instance- und Ressourcenverwaltung), verfügen jedoch nicht über eine Webservice-API für die Arbeit mit den Daten. Das SDK for Java 2.x verfügt über eine entsprechende [DocDbClient](#) Schnittstelle für die Arbeit mit der Ressource. Sie benötigen jedoch eine andere Java-API, z. B. [MongoDB für Java](#), um mit den Daten zu arbeiten.

Verwenden Sie die folgenden Beispiele, um zu erfahren, wie Sie die Service-Clients des SDK für Java 2.x mit den verschiedenen Arten von Datenbanken verwenden.

### Beispiele für Amazon DynamoDB

Arbeiten mit den Daten

SDK service client: [DynamoDBClient](#)

Example: [React/Spring-REST-Anwendung mit DynamoDB](#)

Examples: [Mehrere DynamoDB-Beispiele](#)

SDK service client: [DynamoDBEnhancedClient](#)

Arbeiten mit der Datenbank

SDK service client: [DynamoDBClient](#)

Examples: [CreateTable, ListTables, DeleteTable](#)

## Arbeiten mit den Daten

Example: [React/Spring-REST-Anwendung mit DynamoDB](#)

Examples: [Mehrere DynamoDB-Beispiele](#)  
(names starting with 'Enhanced')

## Arbeiten mit der Datenbank

Weitere [DynamoDB-Beispiele](#) finden Sie im Abschnitt mit geführten Codebeispielen dieses Handbuchs.

## Beispiele für Amazon RDS

Arbeiten mit den Daten	Arbeiten mit der Datenbank
Nicht-SDK-API: JDBC, datenbankspezifische SQL-Variante; Ihr Code verwaltet Datenbankverbindungen oder einen Verbindungspool.	SDK-Service-Client: <a href="#">RdsClient</a>
Beispiel: <a href="#">React/Spring-REST-Anwendung mit MySQL</a>	Beispiele: <a href="#">Mehrere RdsClient Beispiele</a>

## Amazon-Redshift-Beispiele

Arbeiten mit den Daten	Arbeiten mit der Datenbank
SDK-Service-Client: <a href="#">RedshiftDataClient</a>	SDK-Service-Client: <a href="#">RedshiftClient</a>
Beispiele: <a href="#">Mehrere RedshiftDataClient Beispiele</a>	Beispiele: <a href="#">Mehrere RedshiftClient Beispiele</a>
Beispiel: <a href="#">React/Spring-REST-Anwendung mit RedshiftDataClient</a>	

## Beispiele für Amazon Aurora Serverless v1

Arbeiten mit den Daten	Arbeiten mit der Datenbank
SDK-Service-Client: <a href="#">RdsDataClient</a>	SDK-Service-Client: <a href="#">RdsClient</a>
Beispiel: <a href="#">React/Spring-REST-Anwendung mit RdsDataClient</a>	Beispiele: <a href="#">Mehrere RdsClient Beispiele</a>

## Beispiele für Amazon DocumentDB

Arbeiten mit den Daten	Arbeiten mit der Datenbank
Nicht-SDK-API: MongoDB-spezifische Java-Bibliothek (z. B. <a href="#">MongoDB für Java</a> ), Ihr Code verwaltet Datenbankverbindungen oder einen Verbindungspool.	SDK-Service-Client: <a href="#">DocDbClient</a>
Beispiele: <a href="#">DocumentDB (Mongo)-Entwickler handbuch</a> (wählen Sie die Registerkarte „Java“)	

## Arbeiten mit DynamoDB

Dieser Abschnitt enthält Beispiele, die Ihnen zeigen, wie Sie mit [DynamoDB](#) arbeiten. Die Beispiele in diesem Abschnitt verwenden den standardmäßigen Low-Level-DynamoDB-Client ([DynamoDbClient](#)), der mit Version 2.x angeboten wird. AWS SDK for Java

Das SDK bietet auch den [DynamoDB Enhanced Client](#), der einen objektorientierten Ansatz auf hoher Ebene für die Arbeit mit DynamoDB bietet.

### Themen

- [Arbeiten mit Tabellen in DynamoDB](#)
- [Arbeiten mit Elementen in DynamoDB](#)
- [Ordnen Sie Java-Objekte DynamoDB-Elementen zu mit dem AWS SDK for Java 2.x](#)

## Arbeiten mit Tabellen in DynamoDB

Tabellen sind die Container für alle Elemente in einer DynamoDB Datenbank. Bevor Sie Daten hinzufügen oder aus entfernen können DynamoDB, müssen Sie eine Tabelle erstellen.

Für jede Tabelle definieren Sie:

- Ein Tabellename, der für Ihr Konto und Ihre Region eindeutig ist.
- Einen Primärschlüssel, für den jeder Wert eindeutig sein muss. Ihre Tabelle kann keine zwei Elemente mit demselben Primärschlüsselwert enthalten.

Ein Primärschlüssel kann einfach sein, also aus einem Schlüssel mit einer einzigen Partition (HASH) bestehen, oder zusammengesetzt, also aus einer Partition und einem Sortierschlüssel (RANGE).

Jedem Schlüsselwert ist ein Datentyp zugeordnet, der von der [ScalarAttributeType](#) Klasse aufgezählt wird. Der Schlüsselwert kann binär (B), numerisch (n) oder eine Zeichenfolge (S) sein. Weitere Informationen finden Sie unter [Benennungsregeln und Datentypen](#) im - Amazon DynamoDB Entwicklerhandbuch.

- Der bereitgestellte Durchsatz gibt Werte an, die die Anzahl der reservierten Lese-Schreib-Kapazitätseinheiten für die Tabelle angeben.

### Note

Die [-Amazon DynamoDB Preise](#) basieren auf den bereitgestellten Durchsatzwerten, die Sie für Ihre Tabellen festgelegt haben. Reservieren Sie daher nur so viel Kapazität, wie Sie für Ihre Tabelle glauben.

Der bereitgestellte Durchsatz für eine Tabelle kann jederzeit geändert werden. So können Sie die Kapazität anpassen, wenn sich Ihre Anforderungen ändern.

## Erstellen einer Tabelle

Verwenden Sie die `DynamoDbClient.createTable` Methode, um eine neue DynamoDB Tabelle zu erstellen. Sie müssen Tabellenattribute und ein Tabellenschema erstellen. Beide Komponenten fließen in den Primärschlüssel der Tabelle ein. Sie müssen auch anfänglich bereitgestellte Durchsatzwerte und einen Tabellennamen angeben.

**Note**

Wenn bereits eine Tabelle mit dem von Ihnen ausgewählten Namen vorhanden ist, [DynamoDbException](#) wird eine ausgelöst.

## Erstellen einer Tabelle mit einem einfachen Primärschlüssel

Dieser Code erstellt eine Tabelle mit einem Attribut, das der einfache Primärschlüssel der Tabelle ist. Das Beispiel verwendet - [AttributeDefinition](#) und - [KeySchemaElement](#) Objekte für die [CreateTableRequest](#).

### Importe

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;
```

### Code

```
public static String createTable(DynamoDbClient ddb, String tableName, String key)
{
    DynamoDbWaiter dbWaiter = ddb.waiter();
    CreateTableRequest request = CreateTableRequest.builder()
        .attributeDefinitions(AttributeDefinition.builder()
            .attributeName(key)
            .attributeType(ScalarAttributeType.S)
            .build())
        .keySchema(KeySchemaElement.builder()
```

```
        .attributeName(key)
        .keyType(KeyType.HASH)
        .build()
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(new Long(10))
        .writeCapacityUnits(new Long(10))
        .build())
    .tableName(tableName)
    .build();

String newTable = "";
try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created
    WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);

    newTable = response.tableDescription().tableName();
    return newTable;

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Erstellen einer Tabelle mit einem zusammengesetzten Primärschlüssel

Im folgenden Beispiel wird eine Tabelle mit zwei Attributen erstellt. Beide Attribute werden für den zusammengesetzten Primärschlüssel verwendet.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

```
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
```

## Code

```
public static String createTableComKey(DynamoDbClient ddb, String tableName) {
    CreateTableRequest request = CreateTableRequest.builder()
        .attributeDefinitions(
            AttributeDefinition.builder()
                .attributeName("Language")
                .attributeType(ScalarAttributeType.S)
                .build(),
            AttributeDefinition.builder()
                .attributeName("Greeting")
                .attributeType(ScalarAttributeType.S)
                .build())
        .keySchema(
            KeySchemaElement.builder()
                .attributeName("Language")
                .keyType(KeyType.HASH)
                .build(),
            KeySchemaElement.builder()
                .attributeName("Greeting")
                .keyType(KeyType.RANGE)
                .build())
        .provisionedThroughput(
            ProvisionedThroughput.builder()
                .readCapacityUnits(new Long(10))
                .writeCapacityUnits(new Long(10)).build())
        .tableName(tableName)
        .build();

    String tableId = "";

    try {
        CreateTableResponse result = ddb.createTable(request);
        tableId = result.tableDescription().tableId();
    }
```

```
        return tableId;
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Auflisten von Tabellen

Sie können die Tabellen in einer bestimmten Region auflisten, indem Sie die `-DynamoDbClient`'s `listTables` Methode aufrufen.

### Note

Wenn die benannte Tabelle für Ihr Konto und Ihre Region nicht vorhanden ist, [ResourceNotFoundException](#) wird eine ausgelöst.

## Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.List;
```

## Code

```
public static void listAllTables(DynamoDbClient ddb){

    boolean moreTables = true;
    String lastName = null;

    while(moreTables) {
        try {
            ListTablesResponse response = null;
```



```
    if (lastName == null) {
        ListTablesRequest request = ListTablesRequest.builder().build();
        response = ddb.listTables(request);
    } else {
        ListTablesRequest request = ListTablesRequest.builder()
            .exclusiveStartTableName(lastName).build();
        response = ddb.listTables(request);
    }

    List<String> tableNames = response.tableNames();

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

    lastName = response.lastEvaluatedTableName();
    if (lastName == null) {
        moreTables = false;
    }
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
System.out.println("\nDone!");
}
```

Standardmäßig werden bis zu 100 Tabellen pro Aufruf zurückgegeben – verwenden Sie `lastEvaluatedTableName` für das zurückgegebene [ListTablesResponse](#) Objekt, um die letzte ausgewertete Tabelle abzurufen. Mit diesem Wert können Sie die Auflistung nach dem zuletzt zurückgegebenen Wert der vorherigen Auflistung beginnen.

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Beschreiben (Abrufen von Informationen zu) einer Tabelle

Verwenden Sie die `-DynamoDbClient`'s `describeTable` Methode, um Informationen zu einer Tabelle abzurufen.

**Note**

Wenn die benannte Tabelle für Ihr Konto und Ihre Region nicht vorhanden ist, [ResourceNotFoundException](#) wird eine ausgelöst.

**Importe**

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughputDescription;
import software.amazon.awssdk.services.dynamodb.model.TableDescription;
import java.util.List;
```

**Code**

```
public static void describeDynamoDBTable(DynamoDbClient ddb, String tableName ) {

    DescribeTableRequest request = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        TableDescription tableInfo =
            ddb.describeTable(request).table();

        if (tableInfo != null) {
            System.out.format("Table name   : %s\n",
                tableInfo.tableName());
            System.out.format("Table ARN   : %s\n",
                tableInfo.tableArn());
            System.out.format("Status      : %s\n",
                tableInfo.tableStatus());
            System.out.format("Item count  : %d\n",
                tableInfo.itemCount().longValue());
            System.out.format("Size (bytes): %d\n",
                tableInfo.tableSizeBytes().longValue());

            ProvisionedThroughputDescription throughputInfo =
```

```
        tableInfo.provisionedThroughput();
        System.out.println("Throughput");
        System.out.format("  Read Capacity : %d\n",
            throughputInfo.readCapacityUnits().longValue());
        System.out.format("  Write Capacity: %d\n",
            throughputInfo.writeCapacityUnits().longValue());

        List<AttributeDefinition> attributes =
            tableInfo.attributeDefinitions();
        System.out.println("Attributes");

        for (AttributeDefinition a : attributes) {
            System.out.format("  %s (%s)\n",
                a.attributeName(), a.attributeType());
        }
    }
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println("\nDone!");
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Ändern (Aktualisieren) einer Tabelle

Sie können die bereitgestellten Durchsatzwerte Ihrer Tabelle jederzeit ändern, indem Sie die `-DynamoDbClient` `updateTable` Methode aufrufen.

### Note

Wenn die benannte Tabelle für Ihr Konto und Ihre Region nicht vorhanden ist, [ResourceNotFoundException](#) wird eine ausgelöst.

## Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.UpdateTableRequest;
```

```
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

## Code

```
public static void updateDynamoDBTable(DynamoDbClient ddb,
                                       String tableName,
                                       Long readCapacity,
                                       Long writeCapacity) {

    System.out.format(
        "Updating %s with new provisioned throughput values\n",
        tableName);
    System.out.format("Read capacity : %d\n", readCapacity);
    System.out.format("Write capacity : %d\n", writeCapacity);

    ProvisionedThroughput tableThroughput = ProvisionedThroughput.builder()
        .readCapacityUnits(readCapacity)
        .writeCapacityUnits(writeCapacity)
        .build();

    UpdateTableRequest request = UpdateTableRequest.builder()
        .provisionedThroughput(tableThroughput)
        .tableName(tableName)
        .build();

    try {
        ddb.updateTable(request);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    System.out.println("Done!");
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Löschen einer Tabelle

Um eine Tabelle zu löschen, rufen Sie die `DynamoDbClient`'s `deleteTable` Methode auf und geben Sie den Namen der Tabelle an.

**Note**

Wenn die benannte Tabelle für Ihr Konto und Ihre Region nicht vorhanden ist, [ResourceNotFoundException](#) wird eine ausgelöst.

**Importe**

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;
```

**Code**

```
public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

**Weitere Informationen**

- [Richtlinien für das Arbeiten mit Tabellen](#) im - Amazon DynamoDB Entwicklerhandbuch
- [Arbeiten mit Tabellen in DynamoDB](#) im - Amazon DynamoDB Entwicklerhandbuch

## Arbeiten mit Elementen in DynamoDB

In DynamoDB ist ein Element eine Sammlung aus Attributen, von denen jedes über einen Namen und einen Wert verfügt. Ein Attributwert kann eine Skalarfunktion, eine Gruppe oder ein Dokumenttyp sein. Weitere Informationen finden Sie unter [Benennungsregeln und Datentypen](#) im -Amazon DynamoDBEntwicklerhandbuch.

### Abrufen (get) eines Elements aus einer Tabelle

Rufen Sie die `getItem` Methode `DynamoDbClient` auf und übergeben Sie ihm ein [GetItemRequest](#) Objekt mit dem Tabellennamen und dem Primärschlüsselwert des gewünschten Elements. Es gibt ein [GetItemResponse](#) Objekt mit allen Attributen für dieses Element zurück. Sie können einen oder mehrere [Projektionsausdrücke](#) in der `GetItemRequest` angeben, um spezifische Attribute abzurufen.

Sie können die `item()` Methode des zurückgegebenen `GetItemResponse` Objekts verwenden, um eine [Zuordnung](#) von Schlüsselpaaren (Zeichenfolge) und Wertpaaren ([AttributeValue](#)) abzurufen, die dem Element zugeordnet sind.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
```

### Code

```
public static void getDynamoDBItem(DynamoDbClient ddb, String tableName, String
key, String keyVal ) {

    HashMap<String, AttributeValue> keyToGet = new HashMap<String, AttributeValue>();

    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal).build());
```

```
GetItemRequest request = GetItemRequest.builder()
    .key(keyToGet)
    .tableName(tableName)
    .build();

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

    if (returnedItem != null) {
        Set<String> keys = returnedItem.keySet();
        System.out.println("Amazon DynamoDB table attributes: \n");

        for (String key1 : keys) {
            System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
        }
    } else {
        System.out.format("No item found with the key %s!\n", key);
    }
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Abrufen eines Elements aus einer Tabelle mithilfe des asynchronen Clients (get)

Rufen Sie die `getItem`-Methode des `DynamoDbAsyncClient` auf und übergeben Sie ihm ein [GetItemRequest](#) Objekt mit dem Tabellennamen und dem Primärschlüsselwert des gewünschten Elements.

Sie können eine [Collection](#)-Instance mit allen Attributen für dieses Element zurückgeben (siehe das folgende Beispiel).

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
```

```
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

## Code

```
public static void getItem(DynamoDbAsyncClient client, String tableName, String
key, String keyVal) {

    HashMap<String, AttributeValue> keyToGet =
        new HashMap<String, AttributeValue>();

    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal).build());

    try {

        // Create a GetItemRequest instance
        GetItemRequest request = GetItemRequest.builder()
            .key(keyToGet)
            .tableName(tableName)
            .build();

        // Invoke the DynamoDbAsyncClient object's getItem
        java.util.Collection<AttributeValue> returnedItem =
client.getItem(request).join().item().values();

        // Convert Set to Map
        Map<String, AttributeValue> map =
returnedItem.stream().collect(Collectors.toMap(AttributeValue::s, s->s));
        Set<String> keys = map.keySet();
        for (String sinKey : keys) {
            System.out.format("%s: %s\n", sinKey, map.get(sinKey).toString());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.



## Hinzufügen eines neuen Elements in einer Tabelle

Erstellen Sie eine [Map](#) mit Schlüssel-Wert-Paaren, die die Attribute des Elements darstellen. Diese müssen Werte für die Primärschlüsselfelder der Tabelle enthalten. Wenn das Element mit dem Primärschlüssel bereits vorhanden ist, werden dessen Felder durch die Anforderung aktualisiert.

### Note

Wenn die benannte Tabelle für Ihr Konto und Ihre Region nicht vorhanden ist, [ResourceNotFoundException](#) wird eine ausgelöst.

## Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import java.util.HashMap;
```

## Code

```
public static void putItemInTable(DynamoDbClient ddb,
                                String tableName,
                                String key,
                                String keyVal,
                                String albumTitle,
                                String albumTitleValue,
                                String awards,
                                String awardVal,
                                String songTitle,
                                String songTitleVal){

    HashMap<String,AttributeValue> itemValues = new
    HashMap<String,AttributeValue>();

    // Add all content to the table
    itemValues.put(key, AttributeValue.builder().s(keyVal).build());
    itemValues.put(songTitle, AttributeValue.builder().s(songTitleVal).build());
```

```
        itemValues.put(albumTitle,
AttributeValue.builder().s(albumTitleValue).build());
        itemValues.put(awards, AttributeValue.builder().s(awardVal).build());

        PutItemRequest request = PutItemRequest.builder()
            .tableName(tableName)
            .item(itemValues)
            .build();

        try {
            ddb.putItem(request);
            System.out.println(tableName + " was successfully updated");

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

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Aktualisieren eines vorhandenen Elements in einer Tabelle

Sie können ein Attribut für ein Element, das bereits in einer Tabelle vorhanden ist, aktualisieren, indem Sie die `updateItem`-Methode des `DynamoDbClient` aufrufen und einen Tabellennamen, Primärschlüsselwert und eine Zuordnung der Felder übergeben, die aktualisiert werden sollen.

### Note

Wenn die benannte Tabelle für Ihr Konto und Ihre Region nicht vorhanden ist oder wenn das durch den Primärschlüssel identifizierte Element, den Sie übergeben haben, nicht vorhanden ist, [ResourceNotFoundException](#) wird eine ausgelöst.

## Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.AttributeAction;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.AttributeValueUpdate;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.HashMap;
```

## Code

```
public static void updateTableItem(DynamoDbClient ddb,
                                   String tableName,
                                   String key,
                                   String keyVal,
                                   String name,
                                   String updateVal){

    HashMap<String,AttributeValue> itemKey = new HashMap<String,AttributeValue>();

    itemKey.put(key, AttributeValue.builder().s(keyVal).build());

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

    // Update the column specified by name with updatedVal
    updatedValues.put(name, AttributeValueUpdate.builder()
        .value(AttributeValue.builder().s(updateVal).build())
        .action(AttributeAction.PUT)
        .build());

    UpdateItemRequest request = UpdateItemRequest.builder()
        .tableName(tableName)
        .key(itemKey)
        .attributeUpdates(updatedValues)
        .build();

    try {
        ddb.updateItem(request);
    } catch (ResourceNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
    } catch (DynamoDbException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
  
    System.out.println("Done!");  
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Löschen eines vorhandenen Elements aus einer Tabelle

Sie können ein Element löschen, das in einer Tabelle vorhanden ist, indem Sie die `deleteItem`-Methode `DynamoDbClient` verwenden und einen Tabellennamen sowie den Primärschlüsselwert angeben.

### Note

Wenn die benannte Tabelle für Ihr Konto und Ihre Region nicht vorhanden ist oder wenn das durch den Primärschlüssel identifizierte Element, den Sie übergeben haben, nicht vorhanden ist, [ResourceNotFoundException](#) wird eine ausgelöst.

## Importe

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;  
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;  
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;  
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;  
import java.util.HashMap;
```

## Code

```
public static void deleteDynamoDBItem(DynamoDbClient ddb, String tableName, String  
key, String keyVal) {  
  
    HashMap<String, AttributeValue> keyToGet =  
        new HashMap<String, AttributeValue>();  
  
    keyToGet.put(key, AttributeValue.builder()
```

```
        .s(keyVal)
        .build());

DeleteItemRequest deleteReq = DeleteItemRequest.builder()
    .tableName(tableName)
    .key(keyToGet)
    .build();

try {
    ddb.deleteItem(deleteReq);
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Weitere Informationen

- [Richtlinien für das Arbeiten mit Elementen](#) im -Amazon DynamoDBEntwicklerhandbuch
- [Arbeiten mit Elementen in DynamoDB](#) im -Amazon DynamoDBEntwicklerhandbuch

## Ordnen Sie Java-Objekte DynamoDB-Elementen zu mit dem AWS SDK for Java 2.x

Die [DynamoDB Enhanced Client API](#) ist eine High-Level-Bibliothek, die der Nachfolger der DynamoDBMapper Klasse von im SDK for Java v1.x ist. Er bietet eine einfache Möglichkeit, clientseitige Klassen zu DynamoDB-Tabellen zuzuordnen. Sie definieren die Beziehungen zwischen Tabellen und ihren jeweiligen Modellklassen im Code. Nach der Definition dieser Beziehungen können Sie verschiedene Operationen zum Erstellen, Lesen, Aktualisieren oder Löschen (Create, Read, Update, Delete, CRUD) für Tabellen oder Elemente in DynamoDB auf intuitive Weise ausführen.

Die DynamoDB Enhanced Client API umfasst auch die [Enhanced Document API](#), mit der Sie mit dokumentartigen Elementen arbeiten können, die keinem definierten Schema folgen.

Die DynamoDB Enhanced Client API wird in den folgenden Themen behandelt.

- [Erste Schritte mit der DynamoDB Enhanced Client API](#)

- [Grundlagen der DynamoDB Enhanced Client API](#)
- [Verwenden Sie erweiterte Mapping-Funktionen](#)
- [Arbeiten Sie mit JSON-Dokumenten mit der Enhanced Document API für DynamoDB](#)
- [Verwenden Sie Erweiterungen](#)
- [Verwenden Sie die DynamoDB Enhanced Client API asynchron](#)
- [Anmerkungen zu Datenklassen](#)

## Erste Schritte mit der DynamoDB Enhanced Client API

Das folgende Tutorial stellt Ihnen die Grundlagen vor, die Sie für die Arbeit mit der DynamoDB Enhanced Client API benötigen.

### Hinzufügen von Abhängigkeiten

Um mit der DynamoDB Enhanced Client API in Ihrem Projekt zu beginnen, fügen Sie eine Abhängigkeit vom dynamodb-enhanced Maven-Artefakt hinzu. Dies wird in den folgenden Beispielen gezeigt.

### Maven

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

```
</project>
```

Führen Sie eine Suche im zentralen Maven-Repository nach der [neuesten Version](#) durch und ersetzen Sie `<VERSION>` durch diesen Wert.

## Gradle

```
repositories {
    mavenCentral()
}
dependencies {
    implementation(platform("software.amazon.awssdk:bom:<VERSION>"))
    implementation("software.amazon.awssdk:dynamodb-enhanced")
    ...
}
```

Führen Sie eine Suche im zentralen Maven-Repository nach der [neuesten Version](#) durch und ersetzen Sie `<VERSION>` durch diesen Wert.

## Generieren eines **TableSchema** aus einer Datenklasse

Ein [TableSchema](#) ermöglicht es dem erweiterten Client, DynamoDB-Attributwerte Ihren clientseitigen Klassen zuzuordnen. In diesem Tutorial erfahren Sie mehr über TableSchemas, die von einer statischen Datenklasse abgeleitet und mithilfe eines Builders aus Code generiert wurden.

## Verwenden einer kommentierten Datenklasse

Das SDK for Java 2.x enthält eine [Reihe von Anmerkungen](#), die Sie mit einer Datenklasse verwenden können, um schnell eine TableSchema für die Zuordnung Ihrer Klassen zu Tabellen zu generieren.

Erstellen Sie zunächst eine Datenklasse, die der [JavaBean Spezifikation](#) entspricht. Die Spezifikation erfordert, dass eine Klasse über einen öffentlichen Konstruktor ohne Argument und über Getter und Setter für jedes Attribut in der Klasse verfügt. Fügen Sie eine Anmerkung auf Klassenebene ein, um anzugeben, dass es sich bei der Datenklasse um eine handeltDynamoDbBean. Fügen Sie außerdem mindestens eine DynamoDbPartitionKeyAnmerkung zum Getter oder Setter für das Primärschlüsselattribut hinzu.

Sie können [Anmerkungen auf Attributebene](#) auf Getter oder Setter anwenden, aber nicht auf beide.

**Note**

Der Begriff `property` wird normalerweise für einen Wert verwendet, der in einem gekapselt ist `JavaBean`. In diesem Handbuch wird jedoch `attribute` stattdessen der Begriff verwendet, um mit der von `DynamoDB` verwendeten Terminologie konsistent zu sein.

Die folgende `Customer` Klasse zeigt die Anmerkungen, die die Klassendefinition mit der `DynamoDB`-Tabelle verknüpfen.

**Customer-Klasse**

```
package org.example.tests.model;

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbBean;
import
  software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.time.Instant;

@DynamoDbBean
public class Customer {

    private String id;
    private String name;
    private String email;
    private Instant regDate;

    @DynamoDbPartitionKey
    public String getId() { return this.id; }

    public void setId(String id) { this.id = id; }

    public String getCustName() { return this.name; }

    public void setCustName(String name) { this.name = name; }

    @DynamoDbSortKey
    public String getEmail() { return this.email; }

    public void setEmail(String email) { this.email = email; }
```



```

    public Instant getRegistrationDate() { return this.regDate; }

    public void setRegistrationDate(Instant registrationDate) { this.regDate =
registrationDate; }

    @Override
    public String toString() {
        return "Customer [id=" + id + ", name=" + name + ", email=" + email
            + ", regDate=" + regDate + "];"
    }
}

```

Nachdem Sie eine kommentierte Datenklasse erstellt haben, verwenden Sie sie, um die zu erstellenTableSchema, wie im folgenden Ausschnitt gezeigt.

```

static final TableSchema<Customer> customerTableSchema =
    TableSchema.fromBean(Customer.class);

```

Ein TableSchema ist so konzipiert, dass er statisch und unveränderlich ist. Sie können sie normalerweise zum Zeitpunkt des Ladens der Klasse instanziiieren.

Die statische TableSchema.fromBean() Factory-Methode inspektiert den Bean, um die Zuordnung von Datenklassenattributen zu und von DynamoDB-Attributen zu generieren.

Ein Beispiel für die Arbeit mit einem Datenmodell, das aus mehreren Datenklassen besteht, finden Sie in der Person Klasse im [???](#) Abschnitt .

### Verwenden eines Builders

Sie können die Kosten für die Bean-Introspektion überspringen, wenn Sie das Tabellenschema im Code definieren. Wenn Sie das Schema codieren, muss Ihre Klasse weder den JavaBean Benennungsstandards folgen noch mit Anmerkungen versehen werden. Das folgende Beispiel verwendet einen Builder und entspricht dem Customer Klassenbeispiel, das Anmerkungen verwendet.

```

static final TableSchema<Customer> customerTableSchema =
    TableSchema.builder(Customer.class)
        .newItemSupplier(Customer::new)
        .addAttribute(String.class, a -> a.name("id"))
        .getter(Customer::getId)

```

```

        .setter(Customer::setId)
        .tags(StaticAttributeTags.primaryPartitionKey()))
    .addAttribute(String.class, a -> a.name("email")
        .getter(Customer::getEmail)
        .setter(Customer::setEmail)
        .tags(StaticAttributeTags.primarySortKey()))
    .addAttribute(String.class, a -> a.name("name")
        .getter(Customer::getCustName)
        .setter(Customer::setCustName))
    .addAttribute(Instant.class, a -> a.name("registrationDate")
        .getter(Customer::getRegistrationDate)
        .setter(Customer::setRegistrationDate))
    .build();

```

## Erstellen eines erweiterten Clients und **DynamoDbTable**

### Erstellen eines erweiterten Clients

Die [DynamoDbEnhancedClient](#) Klasse oder ihr asynchrones Gegenstück, [DynamoDbEnhancedAsyncClient](#), ist der Einstiegspunkt für die Arbeit mit der DynamoDB Enhanced Client API.

Der erweiterte Client benötigt einen Standard [DynamoDbClient](#) für die Ausführung von Aufgaben. Die API bietet zwei Möglichkeiten, eine `DynamoDbEnhancedClient` Instance zu erstellen. Die erste Option, die im folgenden Ausschnitt gezeigt wird, erstellt einen Standard `DynamoDbClient` mit Standardeinstellungen, die aus den Konfigurationseinstellungen übernommen werden.

```
DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.create();
```

Wenn Sie den zugrunde liegenden Standardclient konfigurieren möchten, können Sie ihn der Builder-Methode des erweiterten Clients bereitstellen, wie im folgenden Codeausschnitt gezeigt.

```

DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
    .dynamoDbClient(
        // Configure an instance of the standard client.
        DynamoDbClient.builder()
            .region(Region.US_EAST_1)

    .credentialsProvider(ProfileCredentialsProvider.create())
        .build())
    .build();

```

## Erstellen einer **DynamoDbTable**-Instance

Stellen Sie sich ein [DynamoDbTable](#) als clientseitige Darstellung einer DynamoDB-Tabelle vor, die die von einem bereitgestellte Mapping-Funktionalität verwendet `TableSchema`. Die `DynamoDbTable` Klasse bietet Methoden für CRUD-Operationen, mit denen Sie mit einer einzelnen DynamoDB-Tabelle interagieren können.

`DynamoDbTable<T>` ist eine generische Klasse, die ein einzelnes Typargument verwendet, unabhängig davon, ob es sich um eine benutzerdefinierte Klasse oder eine `EnhancedDocument` handelt, wenn sie mit Dokumententypen arbeitet. Dieser Argumenttyp stellt die Beziehung zwischen der Klasse, die Sie verwenden, und der einzelnen DynamoDB-Tabelle her.

Verwenden Sie die `table()` Factory-Methode von `DynamoDbEnhancedClient`, um eine `DynamoDbTable` Instance zu erstellen, wie im folgenden Ausschnitt gezeigt.

```
static final DynamoDbTable<Customer> customerTable =
    enhancedClient.table("Customer", TableSchema.fromBean(Customer.class));
```

`DynamoDbTable` -Instances sind Kandidaten für Singletons, da sie unveränderlich sind und in Ihrer gesamten Anwendung verwendet werden können.

Ihr Code enthält jetzt eine speicherinterne Darstellung einer DynamoDB-Tabelle, die `Customer` Instances speichern kann. Die tatsächliche DynamoDB-Tabelle ist möglicherweise vorhanden oder nicht. Wenn die Tabelle mit dem Namen `Customer` bereits vorhanden ist, können Sie damit beginnen, CRUD-Operationen für sie auszuführen. Wenn sie nicht vorhanden ist, verwenden Sie die `DynamoDbTable` Instance, um die Tabelle zu erstellen, wie im nächsten Abschnitt beschrieben.

### Erstellen einer DynamoDB-Tabelle bei Bedarf

Nachdem Sie eine `DynamoDbTable` Instance erstellt haben, verwenden Sie sie, um eine einmalige Erstellung einer Tabelle in DynamoDB durchzuführen.

### Beispielcode für Tabelle erstellen

Im folgenden Beispiel wird eine DynamoDB-Tabelle basierend auf der `Customer` Datenklasse erstellt.

In diesem Beispiel wird eine DynamoDB-Tabelle mit dem Namen `customer` erstellt – Identisch mit dem Klassennamen –, der Tabellename kann jedoch etwas anderes sein. Unabhängig davon,

welchen Namen Sie der Tabelle haben, müssen Sie diesen Namen in zusätzlichen Anwendungen verwenden, um mit der Tabelle zu arbeiten. Geben Sie diesen Namen `table()` immer dann an, wenn Sie ein anderes `DynamoDbTable` Objekt erstellen, um mit der zugrunde liegenden DynamoDB-Tabelle zu arbeiten.

Mit dem Java-Lambda-Parameter , der an die `-createTable` Methode übergeben wird `builder`, können Sie [die Tabelle anpassen](#). In diesem Beispiel ist [der bereitgestellte Durchsatz](#) konfiguriert. Wenn Sie beim Erstellen einer Tabelle Standardeinstellungen verwenden möchten, überspringen Sie den Builder, wie im folgenden Codeausschnitt gezeigt.

```
customerDynamoDbTable.createTable();
```

Wenn Standardeinstellungen verwendet werden, werden keine Werte für den bereitgestellten Durchsatz festgelegt. Stattdessen wird der Abrechnungsmodus für die Tabelle auf [On-Demand](#) festgelegt.

Das Beispiel verwendet auch einen , [DynamoDbWaiter](#) bevor versucht wird, den in der Antwort empfangenen Tabellennamen auszudrucken. Die Erstellung einer Tabelle dauert einige Zeit. Daher bedeutet die Verwendung eines Waiters, dass Sie keine Logik schreiben müssen, die den DynamoDB-Service abfragt, um festzustellen, ob die Tabelle vorhanden ist, bevor die Tabelle verwendet wird.

## Importe

```
import com.example.dynamodb.Customer;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;
```

## Code

```
public static void createCustomerTable(DynamoDbTable<Customer> customerDynamoDbTable,
    DynamoDbClient dynamoDbClient) {
    // Create the DynamoDB table by using the 'customerDynamoDbTable' DynamoDbTable
    instance.
    customerDynamoDbTable.createTable(builder -> builder
```

```

        .provisionedThroughput(b -> b
            .readCapacityUnits(10L)
            .writeCapacityUnits(10L)
            .build())
    );
    // The 'dynamoDbClient' instance that's passed to the builder for the
    DynamoDbWaiter is the same instance
    // that was passed to the builder of the DynamoDbEnhancedClient instance used to
    create the 'customerDynamoDbTable'.
    // This means that the same Region that was configured on the standard
    'dynamoDbClient' instance is used for all service clients.
    try (DynamoDbWaiter waiter =
    DynamoDbWaiter.builder().client(dynamoDbClient).build()) { // DynamoDbWaiter is
    Autocloseable
        ResponseOrException<DescribeTableResponse> response = waiter
            .waitUntilTableExists(builder -> builder.tableName("Customer").build())
            .matched();
        DescribeTableResponse tableDescription = response.response().orElseThrow(
            () -> new RuntimeException("Customer table was not created."));
        // The actual error can be inspected in response.exception()
        logger.info("Customer table was created.");
    }
}

```

### Note

Die Attributnamen einer DynamoDB-Tabelle beginnen mit einem Kleinbuchstaben, wenn die Tabelle aus einer Datenklasse generiert wird. Wenn der Attributname der Tabelle mit einem Großbuchstaben beginnen soll, verwenden Sie die [@DynamoDbAttribute\(NAME\)Anmerkung](#) und geben Sie den gewünschten Namen als Parameter an.

## Ausführen von -Operationen

Nachdem die Tabelle erstellt wurde, verwenden Sie die `DynamoDbTable` Instance, um Operationen für die DynamoDB-Tabelle auszuführen.

Im folgenden Beispiel `DynamoDbTable<Customer>` wird ein Singleton als Parameter zusammen mit einer [Customer Datenklassen](#)-Instance übergeben, um der Tabelle ein neues Element hinzuzufügen.

```
public static void putItemExample(DynamoDbTable<Customer> customerTable, Customer
customer){
    logger.info(customer.toString());
    customerTable.putItem(customer);
}
```

## Customer-Objekt

```
Customer customer = new Customer();
customer.setId("1");
customer.setCustName("Customer Name");
customer.setEmail("customer@example.com");
customer.setRegistrationDate(Instant.parse("2023-07-03T10:15:30.00Z"));
```

Bevor Sie das `customer` Objekt an den DynamoDB-Service senden, protokollieren Sie die Ausgabe der `toString()` Objektmethode, um sie mit dem zu vergleichen, was der erweiterte Client sendet.

```
Customer [id=1, name=Customer Name, email=customer@example.com,
regDate=2023-07-03T10:15:30Z]
```

Die Protokollierung auf Wire-Ebene zeigt die Nutzlast der generierten Anforderung an. Der erweiterte Client hat die Low-Level-Darstellung aus der Datenklasse generiert. Das `regDate` Attribut, bei dem es sich um einen `-InstantTyp` in Java handelt, wird als DynamoDB-Zeichenfolge dargestellt.

```
{
  "TableName": "Customer",
  "Item": {
    "registrationDate": {
      "S": "2023-07-03T10:15:30Z"
    },
    "id": {
      "S": "1"
    },
    "custName": {
      "S": "Customer Name"
    },
    "email": {
      "S": "customer@example.com"
    }
  }
}
```

```
}
```

## Arbeiten mit einer vorhandenen Tabelle

Im vorherigen Abschnitt wurde gezeigt, wie eine DynamoDB-Tabelle erstellt wird, die mit einer Java-Datenklasse beginnt. Wenn Sie bereits über eine vorhandene Tabelle verfügen und die Funktionen des erweiterten Clients verwenden möchten, können Sie eine Java-Datenklasse für die Arbeit mit der Tabelle erstellen. Sie müssen die DynamoDB-Tabelle untersuchen und der Datenklasse die erforderlichen Anmerkungen hinzufügen.

Bevor Sie mit einer vorhandenen Tabelle arbeiten, rufen Sie die `-DynamoDbEnhanced.table()` Methode auf. Dies wurde im vorherigen Beispiel mit der folgenden Anweisung geschehen.

```
DynamoDbTable<Customer> customerTable = enhancedClient.table("Customer",  
    TableSchema.fromBean(Customer.class));
```

Nachdem die `DynamoDbTable` Instance zurückgegeben wurde, können Sie sofort mit der zugrunde liegenden Tabelle arbeiten. Sie müssen die Tabelle nicht neu erstellen, indem Sie die `-DynamoDbTable.createTable()` Methode aufrufen.

Das folgende Beispiel zeigt dies, indem sofort eine `Customer` Instance aus der DynamoDB-Tabelle abgerufen wird.

```
DynamoDbTable<Customer> customerTable = enhancedClient.table("Customer",  
    TableSchema.fromBean(Customer.class));  
// The Customer table exists already and has an item with a primary key value of "1"  
// and a sort key value of "customer@example.com".  
customerTable.getItem(  
    Key.builder().  
        partitionValue("1").  
        sortValue("customer@example.com").build());
```

### Important

Der in der `table()` Methode verwendete Tabellename muss mit dem vorhandenen DynamoDB-Tabellennamen übereinstimmen.

## Grundlagen der DynamoDB Enhanced Client API

In diesem Thema werden die grundlegenden Funktionen der DynamoDB Enhanced Client API beschrieben und mit der [standardmäßigen DynamoDB-Client-API](#) verglichen.

Wenn Sie noch nicht mit der DynamoDB Enhanced Client API vertraut sind, empfehlen wir Ihnen, das [einführende Tutorial](#) zu durchlaufen, um sich mit den grundlegenden Klassen vertraut zu machen.

### DynamoDB-Elemente in Java

DynamoDB-Tabellen speichern Elemente. Je nach Anwendungsfall können Elemente auf der Java-Seite die Form statisch strukturierter Daten oder dynamisch erstellter Strukturen annehmen.

Wenn Ihr Anwendungsfall Elemente mit einem konsistenten Satz von Attributen aufruft, verwenden Sie [kommentierte Klassen](#) oder verwenden Sie einen [Builder](#), um den entsprechenden statisch typisierten zu generieren `TableSchema`.

Wenn Sie Elemente speichern müssen, die aus unterschiedlichen Strukturen bestehen, erstellen Sie alternativ eine `DocumentTableSchema`. `DocumentTableSchema` ist Teil der [Enhanced Document API](#) und erfordert nur einen statisch typisierten Primärschlüssel und arbeitet mit `EnhancedDocument` Instances zusammen, um die Datenelemente zu speichern. Die Enhanced Document API wird in einem anderen [Thema behandelt](#).

### Attributtypen

Obwohl DynamoDB im Vergleich zum umfangreichen Java-System [eine kleine Anzahl von Attributtypen](#) unterstützt, bietet die DynamoDB Enhanced Client API Mechanismen zum Konvertieren von Mitgliedern einer Java-Klasse zu und von DynamoDB-Attributtypen.

Standardmäßig unterstützt die DynamoDB Enhanced Client API Attributkonverter für eine große Anzahl von Typen, z. B. [Ganzzahl](#), [Zeichenfolge](#) [BigDecimal](#), und [Instant](#). Die Liste wird in den [bekannten Implementierungsklassen der AttributeConverter Schnittstelle](#) angezeigt. Die Liste enthält viele Typen und Sammlungen wie Karten, Listen und Sätze.

Um die Daten für einen Attributtyp zu speichern, der standardmäßig nicht unterstützt wird oder nicht der JavaBean Konvention entspricht, können Sie eine benutzerdefinierte `AttributeConverter` Implementierung schreiben, um die Konvertierung durchzuführen. Ein [Beispiel finden Sie im Abschnitt Attributkonvertierung](#).

Um die Daten für einen Attributtyp zu speichern, dessen Klasse der Java-Beans-Spezifikation entspricht (oder einer [unveränderlichen Datenklasse](#)), können Sie zwei Ansätze verfolgen.



- Wenn Sie Zugriff auf die Quelldatei haben, können Sie die Klasse mit `@DynamoDbBean` (oder ) kommentieren `@DynamoDbImmutable`. Der Abschnitt, der verschachtelte Attribute behandelt, zeigt [Beispiele](#) für die Verwendung von kommentierten Klassen.
- Wenn Sie keinen Zugriff auf die Quelldatei der JavaBean Datenklasse für das Attribut haben (oder die Quelldatei einer Klasse, auf die Sie Zugriff haben, nicht kommentieren möchten), können Sie den Builder-Ansatz verwenden. Dadurch wird ein Tabellenschema erstellt, ohne die Schlüssel zu definieren. Anschließend können Sie dieses Tabellenschema in ein anderes Tabellenschema verschachteln, um die Zuordnung durchzuführen. Der Abschnitt Verschachteltes Attribut enthält ein [Beispiel](#) für die Verwendung verschachtelter Schemata.

## Werte des primitiven Java-Typs

Obwohl der erweiterte Client mit Attributen primitiver Typen arbeiten kann, empfehlen wir die Verwendung von Objekttypen, da Sie bei primitiven Typen keine Nullwerte darstellen können.

## Null-Werte

Wenn Sie die `putItem` API verwenden, schließt der erweiterte Client keine nullwertigen Attribute eines zugeordneten Datenobjekts in die Anforderung an DynamoDB ein.

Bei `-updateItem`Anforderungen werden nullwertige Attribute aus dem Element in der Datenbank entfernt. Wenn Sie beabsichtigen, einige Attributwerte zu aktualisieren und die anderen unverändert zu lassen, kopieren Sie entweder die Werte anderer Attribute, die nicht geändert werden sollten, oder verwenden Sie die Methode [ignoreNull \(\)](#) im Update Builder.

Das folgende Beispiel zeigt `ignoreNulls()` die `-the updateItem()` Methode.

```
public void updateItemNullsExample(){
    Customer customer = new Customer();
    customer.setCustName("CustName");
    customer.setEmail("email");
    customer.setId("1");
    customer.setRegistrationDate(Instant.now());

    // Put item with values for all attributes.
    customerDynamoDbTable.putItem(customer);

    // Create a Customer instance with the same id value, but a different name
    value.
```

```
// Do not set the 'registrationDate' attribute.
Customer custForUpdate = new Customer();
custForUpdate.setCustName("NewName");
custForUpdate.setEmail("email");
custForUpdate.setId("1");

// Update item without setting the registrationDate attribute.
customerDynamoDbTable.updateItem(b -> b
    .item(custForUpdate)
    .ignoreNulls(Boolean.TRUE));

Customer updatedWithNullsIgnored = customerDynamoDbTable.getItem(customer);
// registrationDate value is unchanged.
logger.info(updatedWithNullsIgnored.toString());

customerDynamoDbTable.updateItem(custForUpdate);
Customer updatedWithNulls = customerDynamoDbTable.getItem(customer);
// registrationDate value is null because ignoreNulls() was not used.
logger.info(updatedWithNulls.toString());
}
}

// Logged lines.
Customer [id=1, custName=NewName, email=email,
  registrationDate=2023-04-05T16:32:32.056Z]
Customer [id=1, custName=NewName, email=email, registrationDate=null]
```

## Grundlegende DynamoDB Enhanced Client-Methoden

Grundlegende Methoden des erweiterten Clients werden den DynamoDB-Service-Operationen zugeordnet, nach denen sie benannt sind. Die folgenden Beispiele zeigen die einfachste Variante jeder Methode. Sie können jede Methode anpassen, indem Sie ein erweitertes Anforderungsobjekt übergeben. Erweiterte Anforderungsobjekte bieten die meisten Funktionen, die im standardmäßigen DynamoDB-Client verfügbar sind. Sie sind vollständig in der AWS SDK for Java 2.x -API-Referenz dokumentiert.

Das Beispiel verwendet das zuvor [the section called “Customer-Klasse”](#) gezeigte .

```
// CreateTable
customerTable.createTable();

// GetItem
```

```
Customer customer =
    customerTable.getItem(Key.builder().partitionValue("a123").build());

// UpdateItem
Customer updatedCustomer = customerTable.updateItem(customer);

// PutItem
customerTable.putItem(customer);

// DeleteItem
Customer deletedCustomer =
    customerTable.deleteItem(Key.builder().partitionValue("a123").sortValue(456).build());

// Query
PageIterable<Customer> customers = customerTable.query(keyEqualTo(k ->
    k.partitionValue("a123")));

// Scan
PageIterable<Customer> customers = customerTable.scan();

// BatchGetItem
BatchGetResultPageIterable batchResults =
    enhancedClient.batchGetItem(r -> r.addReadBatch(ReadBatch.builder(Customer.class)
        .mappedTableResource(customerTable)
        .addGetItem(key1)
        .addGetItem(key2)
        .addGetItem(key3)
        .build()));

// BatchWriteItem
batchResults = enhancedClient.batchWriteItem(r ->
    r.addWriteBatch(WriteBatch.builder(Customer.class)
        .mappedTableResource(customerTable)
        .addPutItem(customer)
        .addDeleteItem(key1)
        .addDeleteItem(key1)
        .build()));

// TransactGetItems
transactResults = enhancedClient.transactGetItems(r -> r.addGetItem(customerTable,
    key1)
        .addGetItem(customerTable,
    key2));
```

```
// TransactWriteItems
enhancedClient.transactWriteItems(r -> r.addConditionCheck(customerTable,
                                                                    i -> i.key(orderKey))

    .conditionExpression(conditionExpression))
                                                                    .addUpdateItem(customerTable, customer)
                                                                    .addDeleteItem(customerTable, key));
```

## Vergleich von DynamoDB Enhanced Client mit dem Standard-DynamoDB-Client

Sowohl DynamoDB-Client-APIs – [Standard](#)- als auch [erweiterte](#) – ermöglichen Ihnen die Arbeit mit DynamoDB-Tabellen, CRUD-Operationen (Erstellen, Lesen, Aktualisieren und Löschen) auf Datenebene durchzuführen. Der Unterschied zwischen den Client-APIs besteht darin, wie dies erreicht wird. Mit dem Standard-Client arbeiten Sie direkt mit Low-Level-Datenattributen. Die erweiterte Client-API verwendet vertraute Java-Klassen und ist der Low-Level-API im Hintergrund zugeordnet.

Während beide Client-APIs Vorgänge auf Datenebene unterstützen, unterstützt der standardmäßige DynamoDB-Client auch Vorgänge auf Ressourcenebene. Operationen auf Ressourcenebene verwalten die Datenbank, z. B. das Erstellen von Backups, das Auflisten von Tabellen und das Aktualisieren von Tabellen. Die erweiterte Client-API unterstützt eine ausgewählte Anzahl von Operationen auf Ressourcenebene, wie das Erstellen, Beschreiben und Löschen von Tabellen.

Um die verschiedenen Ansätze zu veranschaulichen, die von den beiden Client-APIs verwendet werden, zeigen die folgenden Codebeispiele die Erstellung derselben ProductCatalog Tabelle mit dem Standardclient und dem erweiterten Client.

## Vergleich: Erstellen einer Tabelle mit dem DynamoDB-Standardclient

```
DependencyFactory.dynamoDbClient().createTable(builder -> builder
    .tableName(TABLE_NAME)
    .attributeDefinitions(
        b -> b.attributeName("id").attributeType(ScalarAttributeType.N),
        b -> b.attributeName("title").attributeType(ScalarAttributeType.S),
        b -> b.attributeName("isbn").attributeType(ScalarAttributeType.S)
    )
    .keySchema(
        builder1 -> builder1.attributeName("id").keyType(KeyType.HASH),
        builder2 -> builder2.attributeName("title").keyType(KeyType.RANGE)
    )
    .globalSecondaryIndexes(builder3 -> builder3
        .indexName("products_by_isbn"))
```

```

        .keySchema(builder2 -> builder2
            .attributeName("isbn").keyType(KeyType.HASH))
        .projection(builder2 -> builder2
            .projectionType(ProjectionType.INCLUDE)
            .nonKeyAttributes("price", "authors"))
        .provisionedThroughput(builder4 -> builder4
            .writeCapacityUnits(5L).readCapacityUnits(5L))
    )
    .provisionedThroughput(builder1 -> builder1
        .readCapacityUnits(5L).writeCapacityUnits(5L))
);

```

## Vergleich: Erstellen einer Tabelle mit dem DynamoDB Enhanced Client

```

DynamoDbEnhancedClient enhancedClient = DependencyFactory.enhancedClient();
productCatalog = enhancedClient.table(TABLE_NAME,
    TableSchema.fromImmutableClass(ProductCatalog.class));
productCatalog.createTable(b -> b
    .provisionedThroughput(b1 -> b1.readCapacityUnits(5L).writeCapacityUnits(5L))
    .globalSecondaryIndices(b2 -> b2.indexName("products_by_isbn")
        .projection(b4 -> b4
            .projectionType(ProjectionType.INCLUDE)
            .nonKeyAttributes("price", "authors"))
        .provisionedThroughput(b3 ->
            b3.writeCapacityUnits(5L).readCapacityUnits(5L))
    )
);

```

Der erweiterte Client verwendet die folgende kommentierte Datenklasse. Der DynamoDB Enhanced Client ordnet DynamoDB-Datentypen Java-Datentypen für weniger ausführlichen Code zu, der leichter zu folgen ist. `ProductCatalog` ist ein Beispiel für die Verwendung einer unveränderlichen Klasse mit dem DynamoDB Enhanced Client. Die Verwendung unveränderlicher Klassen für zugeordnete Datenklassen wird [weiter unten in diesem Thema erläutert](#).

### ProductCatalog-Klasse

```

package org.example.tests.model;

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbIgnore;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbImmutable;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;

```

```
import
software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.math.BigDecimal;
import java.util.Objects;
import java.util.Set;

@DynamoDbImmutable(builder = ProductCatalog.Builder.class)
public class ProductCatalog implements Comparable<ProductCatalog> {
    private Integer id;
    private String title;
    private String isbn;
    private Set<String> authors;
    private BigDecimal price;

    private ProductCatalog(Builder builder){
        this.authors = builder.authors;
        this.id = builder.id;
        this.isbn = builder.isbn;
        this.price = builder.price;
        this.title = builder.title;
    }

    public static Builder builder(){ return new Builder(); }

    @DynamoDbPartitionKey
    public Integer id() { return id; }

    @DynamoDbSortKey
    public String title() { return title; }

    @DynamoDbSecondaryPartitionKey(indexNames = "products_by_isbn")
    public String isbn() { return isbn; }
    public Set<String> authors() { return authors; }
    public BigDecimal price() { return price; }

    public static final class Builder {
        private Integer id;
        private String title;
        private String isbn;
        private Set<String> authors;
```

```
private BigDecimal price;
private Builder(){

public Builder id(Integer id) { this.id = id; return this; }
public Builder title(String title) { this.title = title; return this; }
public Builder isbn(String ISBN) { this.isbn = ISBN; return this; }
public Builder authors(Set<String> authors) { this.authors = authors; return
this; }
public Builder price(BigDecimal price) { this.price = price; return this; }
public ProductCatalog build() { return new ProductCatalog(this); }
}

@Override
public String toString() {
    final StringBuffer sb = new StringBuffer("ProductCatalog{");
    sb.append("id=").append(id);
    sb.append(", title=").append(title).append('\ ');
    sb.append(", isbn=").append(isbn).append('\ ');
    sb.append(", authors=").append(authors);
    sb.append(", price=").append(price);
    sb.append('}');
    return sb.toString();
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    ProductCatalog that = (ProductCatalog) o;
    return id.equals(that.id) && title.equals(that.title) && Objects.equals(isbn,
that.isbn) && Objects.equals(authors, that.authors) && Objects.equals(price,
that.price);
}

@Override
public int hashCode() {
    return Objects.hash(id, title, isbn, authors, price);
}

@Override
@DynamoDbIgnore
public int compareTo(ProductCatalog other) {
    if (this.id.compareTo(other.id) != 0){
        return this.id.compareTo(other.id);
    }
}
```

```
    } else {  
        return this.title.compareTo(other.title);  
    }  
}  
}
```

Die folgenden beiden Codebeispiele für einen Batch-Write veranschaulichen die Ausführlichkeit und fehlende Typsicherheit bei Verwendung des Standard-Clients im Gegensatz zum erweiterten Client.

### Vergleich: Batch-Schreibvorgänge mit dem DynamoDB-Standardclient

```
public static void batchWriteStandard(DynamoDbClient dynamoDbClient, String  
tableName) {  
  
    Map<String, AttributeValue> catalogItem = Map.of(  
        "authors", AttributeValue.builder().ss("a", "b").build(),  
        "id", AttributeValue.builder().n("1").build(),  
        "isbn", AttributeValue.builder().s("1-565-85698").build(),  
        "title", AttributeValue.builder().s("Title 1").build(),  
        "price", AttributeValue.builder().n("52.13").build());  
  
    Map<String, AttributeValue> catalogItem2 = Map.of(  
        "authors", AttributeValue.builder().ss("a", "b", "c").build(),  
        "id", AttributeValue.builder().n("2").build(),  
        "isbn", AttributeValue.builder().s("1-208-98073").build(),  
        "title", AttributeValue.builder().s("Title 2").build(),  
        "price", AttributeValue.builder().n("21.99").build());  
  
    Map<String, AttributeValue> catalogItem3 = Map.of(  
        "authors", AttributeValue.builder().ss("g", "k", "c").build(),  
        "id", AttributeValue.builder().n("3").build(),  
        "isbn", AttributeValue.builder().s("7-236-98618").build(),  
        "title", AttributeValue.builder().s("Title 3").build(),  
        "price", AttributeValue.builder().n("42.00").build());  
  
    Set<WriteRequest> writeRequests = Set.of(  
        WriteRequest.builder().putRequest(b -> b.item(catalogItem)).build(),  
        WriteRequest.builder().putRequest(b -> b.item(catalogItem2)).build(),  
        WriteRequest.builder().putRequest(b -> b.item(catalogItem3)).build());  
  
    Map<String, Set<WriteRequest>> productCatalogItems = Map.of(  
        "ProductCatalog", writeRequests);  
}
```



```

    BatchWriteItemResponse response = dynamoDbClient.batchWriteItem(b ->
b.requestItems(productCatalogItems));

    logger.info("Unprocessed items: " + response.unprocessedItems().size());
}

```

## Vergleich: Batch-Write mit dem DynamoDB Enhanced Client

```

public static void batchWriteEnhanced(DynamoDbTable<ProductCatalog> productCatalog)
{
    ProductCatalog prod = ProductCatalog.builder()
        .id(1)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))
        .price(BigDecimal.valueOf(52.13))
        .title("Title 1")
        .build();
    ProductCatalog prod2 = ProductCatalog.builder()
        .id(2)
        .isbn("1-208-98073")
        .authors(new HashSet<>(Arrays.asList("a", "b", "c")))
        .price(BigDecimal.valueOf(21.99))
        .title("Title 2")
        .build();
    ProductCatalog prod3 = ProductCatalog.builder()
        .id(3)
        .isbn("7-236-98618")
        .authors(new HashSet<>(Arrays.asList("g", "k", "c")))
        .price(BigDecimal.valueOf(42.00))
        .title("Title 3")
        .build();

    BatchWriteResult batchWriteResult = DependencyFactory.enhancedClient()
        .batchWriteItem(b -> b.writeBatches(
            WriteBatch.builder(ProductCatalog.class)
                .mappedTableResource(productCatalog)
                .addPutItem(prod).addPutItem(prod2).addPutItem(prod3)
                .build()
        ));
    logger.info("Unprocessed items: " +
batchWriteResult.unprocessedPutItemsForTable(productCatalog).size());
}

```

## Arbeiten mit unveränderlichen Datenklassen

Die Zuordnungsfunktion der DynamoDB Enhanced Client API funktioniert mit unveränderlichen Datenklassen. Eine unveränderliche Klasse hat nur Getter und erfordert eine Builder-Klasse, die das SDK zum Erstellen von Instances der Klasse verwendet. Anstatt die `@DynamoDbBean`-Anmerkung zu verwenden, wie in der [Kundenklasse gezeigt, verwenden unveränderliche Klassen die `@DynamoDbImmutable`-Anmerkung](#), die einen Parameter verwendet, der die zu verwendende Builder-Klasse angibt.

Die folgende Klasse ist eine unveränderliche Version von Customer.

```
package org.example.tests.model.immutable;

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbImmutable;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondarySortKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.time.Instant;

@DynamoDbImmutable(builder = CustomerImmutable.Builder.class)
public class CustomerImmutable {
    private final String id;
    private final String name;
    private final String email;
    private final Instant regDate;

    private CustomerImmutable(Builder b) {
        this.id = b.id;
        this.email = b.email;
        this.name = b.name;
        this.regDate = b.regDate;
    }

    // This method will be automatically discovered and used by the TableSchema.
    public static Builder builder() { return new Builder(); }

    @DynamoDbPartitionKey
    public String id() { return this.id; }
```

```
@DynamoDbSortKey
public String email() { return this.email; }

@DynamoDbSecondaryPartitionKey(indexNames = "customers_by_name")
public String name() { return this.name; }

@DynamoDbSecondarySortKey(indexNames = {"customers_by_date", "customers_by_name"})
public Instant regDate() { return this.regDate; }

public static final class Builder {
    private String id;
    private String email;
    private String name;
    private Instant regDate;

    // The private Builder constructor is visible to the enclosing Customer class.
    private Builder() {}

    public Builder id(String accountId) { this.id = id; return this; }
    public Builder email(String email) { this.email = email; return this; }
    public Builder name(String name) { this.name = name; return this; }
    public Builder regDate(Instant regDate) { this.regDate = regDate; return
this; }

    // This method will be automatically discovered and used by the TableSchema.
    public CustomerImmutable build() { return new CustomerImmutable(this); }
}
}
```

Sie müssen die folgenden Anforderungen erfüllen, wenn Sie eine Datenklasse mit `kommentieren@DynamoDbImmutable`.

1. Jede Methode, bei der es sich nicht um eine Überschreibung von `Object.class` und die nicht mit `annotiert` wurde, `@DynamoDbIgnore` muss ein Getter für ein Attribut der DynamoDB-Tabelle sein.
2. Jeder Getter muss in der Builder-Klasse über einen entsprechenden Setter verfügen, bei dem die Groß- und Kleinschreibung beachtet wird.
3. Es muss nur eine der folgenden Konstruktionsbedingungen erfüllt sein.
  - Die Builder-Klasse muss über einen öffentlichen Standardkonstruktor verfügen.

- Die Datenklasse muss über eine öffentliche statische Methode namens `verfügenbuilder()`, die keine Parameter verwendet und eine Instance der Builder-Klasse zurückgibt. Diese Option wird in der unveränderlichen `Customer` Klasse angezeigt.
4. Die Builder-Klasse muss über eine öffentliche Methode namens `verfügenbuild()`, die keine Parameter verwendet und eine Instance der unveränderlichen Klasse zurückgibt.

Um einen `TableSchema` für Ihre unveränderliche Klasse zu erstellen, verwenden Sie die `-fromImmutableClass()` Methode auf `TableSchema` wie im folgenden Codeausschnitt gezeigt.

```
static final TableSchema<CustomerImmutable> customerImmutableTableSchema =  
    TableSchema.fromImmutableClass(CustomerImmutable.class);
```

So wie Sie eine DynamoDB-Tabelle aus einer veränderlichen Klasse erstellen können, können Sie eine aus einer unveränderlichen Klasse mit einem einmaligen Aufruf `createTable()` von `createTable()` erstellen, `DynamoDbTable` wie im folgenden Beispiel für einen Ausschnitt gezeigt.

```
static void createTableFromImmutable(DynamoDbEnhancedClient enhancedClient, String  
    tableName, DynamoDbWaiter waiter){  
    // First, create an in-memory representation of the table using the 'table()'   
    // method of the DynamoDb Enhanced Client.  
    // 'table()' accepts a name for the table and a TableSchema instance that you   
    // created previously.  
    DynamoDbTable<CustomerImmutable> customerDynamoDbTable = enhancedClient  
        .table(tableName, TableSchema.fromImmutableClass(CustomerImmutable.class));  
  
    // Second, call the 'createTable()' method on the DynamoDbTable instance.  
    customerDynamoDbTable.createTable();  
    waiter.waitUntilTableExists(b -> b.tableName(tableName));  
}
```

## Verwenden von Bibliotheken von Drittanbietern wie Lombok

Bibliotheken von Drittanbietern wie [Project Lombok](#) helfen bei der Generierung von Boilerplate-Code, der unveränderlichen Objekten zugeordnet ist. Die DynamoDB Enhanced Client API funktioniert mit diesen Bibliotheken, solange die Datenklassen den in diesem Abschnitt beschriebenen Konventionen entsprechen.

Das folgende Beispiel zeigt die unveränderliche `CustomerImmutable` Klasse mit Lombok-Anmerkungen. Beachten Sie, wie die `onMethod` Funktion von Lombok attributbasierte DynamoDB-Anmerkungen, wie z. B. `@DynamoDbPartitionKey`, in den generierten Code kopiert.

```
@Value
@Builder
@dynamoDbImmutable(builder = Customer.CustomerBuilder.class)
public class Customer {
    @Getter(onMethod_=@DynamoDbPartitionKey)
    private String id;

    @Getter(onMethod_=@DynamoDbSortKey)
    private String email;

    @Getter(onMethod_=@DynamoDbSecondaryPartitionKey(indexNames = "customers_by_name"))
    private String name;

    @Getter(onMethod_=@DynamoDbSecondarySortKey(indexNames = {"customers_by_date",
"customers_by_name"}))
    private Instant createdAt;
}
```

Verwenden Sie Ausdrücke und Bedingungen

Ausdrücke in der DynamoDB Enhanced Client API sind Java-Repräsentationen von [DynamoDB-Ausdrücken](#).

Die DynamoDB Enhanced Client API verwendet drei Arten von Ausdrücken:

### [Expression](#)

Die `Expression` Klasse wird verwendet, wenn Sie Bedingungen und Filter definieren.

### [QueryConditional](#)

Dieser Ausdruckstyp stellt [wichtige Bedingungen](#) für Abfrageoperationen dar.

### [UpdateExpression](#)

Diese Klasse hilft Ihnen beim Schreiben von [DynamoDB-Aktualisierungsausdrücken](#) und wird derzeit im Erweiterungs-Framework verwendet, wenn Sie ein Element aktualisieren.

## Anatomie des Ausdrucks

Ein Ausdruck setzt sich wie folgt zusammen:

- Ein Zeichenkettenausdruck (erforderlich). Die Zeichenfolge enthält einen DynamoDB-Logikausdruck mit Platzhalternamen für Attributnamen und Attributwerte.
- Eine Zuordnung von Ausdruckswerten (normalerweise erforderlich).
- Eine Zuordnung von Ausdrucksnamen (optional).

Verwenden Sie einen Builder, um ein Expression Objekt zu generieren, das die folgende allgemeine Form annimmt.

```
Expression expression = Expression.builder()
    .expression(<String>)
    .expressionNames(<Map>)
    .expressionValues(<Map>)
    .build()
```

Expressions erfordern normalerweise eine Zuordnung von Ausdruckswerten. Die Map stellt die Werte für die Platzhalter im Zeichenkettenausdruck bereit. Der Map-Schlüssel besteht aus dem Platzhalternamen, dem ein Doppelpunkt (:) vorangestellt ist, und der Zuordnungswert ist eine Instanz von [AttributeValue](#). Die [AttributeValues](#)-Klasse verfügt über praktische Methoden zum Generieren einer [AttributeValue](#)-Instanz aus einem Literal. Alternativ können Sie die verwenden, [AttributeValue.Builder](#) um eine [AttributeValue](#)-Instanz zu generieren.

Das folgende Snippet zeigt eine Map mit zwei Einträgen nach Kommentarzeile 2. Die an die `expression()`-Methode übergebene Zeichenfolge, die nach Kommentarzeile 1 angezeigt wird, enthält die Platzhalter, die DynamoDB vor der Ausführung des Vorgangs auflöst. Dieser Ausschnitt enthält keine Zuordnung von Ausdrucksnamen, da `Preis` ein zulässiger Attributname ist.

```
public static void scanAsync(DynamoDbAsyncTable productCatalog) {
    ScanEnhancedRequest request = ScanEnhancedRequest.builder()
        .consistentRead(true)
        .attributesToProject("id", "title", "authors", "price")
        .filterExpression(Expression.builder()
            // 1. :min_value and :max_value are placeholders for the values
            // provided by the map
            .expression("price >= :min_value AND price <= :max_value")
            // 2. Two values are needed for the expression and each is
            // supplied as a map entry.
```

```

        .expressionValues(
            Map.of( ":min_value", numberValue(8.00),
                  ":max_value", numberValue(400_000.00)))
        .build())
    .build();

```

Wenn ein Attributname in der DynamoDB-Tabelle ein reserviertes Wort ist, mit einer Zahl beginnt oder ein Leerzeichen enthält, ist eine Zuordnung von Ausdrucksnamen erforderlich. Expression

Wenn der Attributname beispielsweise *1price* anstelle von *price* im vorherigen Codebeispiel verwendet wurde, müsste das Beispiel wie im folgenden Beispiel geändert werden.

```

ScanEnhancedRequest request = ScanEnhancedRequest.builder()
    .filterExpression(Expression.builder()
        .expression("#price >= :min_value AND #price <= :max_value")
        .expressionNames( Map.of("#price", "1price") )
        .expressionValues(
            Map.of(":min_value", numberValue(8.00),
                  ":max_value", numberValue(400_000.00)))
        .build())
    .build();

```

Ein Platzhalter für einen Ausdrucksnamen beginnt mit dem Rautenzeichen (#). Ein Eintrag für die Zuordnung von Ausdrucksnamen verwendet den Platzhalter als Schlüssel und den Attributnamen als Wert. Die Map wird dem Ausdrucks-Generator mit der `expressionNames()` Methode hinzugefügt. DynamoDB löst den Attributnamen auf, bevor es den Vorgang ausführt.

Ausdruckswerte sind nicht erforderlich, wenn eine Funktion im Zeichenkettenausdruck verwendet wird. Ein Beispiel für eine Ausdrucksfunktion ist `attribute_exists(<attribute_name>)`.

Im folgenden Beispiel wird eine `expression` erstellt, die eine [DynamoDB-Funktion](#) verwendet. Die Ausdruckszeichenfolge in diesem Beispiel verwendet keine Platzhalter. Dieser Ausdruck könnte bei einer `putItem` Operation verwendet werden, um zu überprüfen, ob in der Datenbank bereits ein Element vorhanden ist, dessen Wert dem `movie` Attribut des Datenobjekts `movie` entspricht.

```

Expression exp = Expression.builder().expression("attribute_not_exists
(movie)").build();

```

Das DynamoDB Developer Guide enthält vollständige Informationen zu den [Low-Level-Ausdrücken, die mit DynamoDB verwendet werden](#).

## Bedingungsausdrücke und Bedingungen

Wenn Sie die `deleteItem()` Methoden `putItem()` `updateItem()`, und sowie Transaktions- und Batchoperationen verwenden, verwenden Sie [Expression](#) Objekte, um Bedingungen anzugeben, die DynamoDB erfüllen muss, um mit dem Vorgang fortzufahren. Diese Ausdrücke sind benannte Bedingungsausdrücke. Ein Beispiel finden Sie in dem Bedingungsdruck, der in der `addDeleteItem()` Methode (nach Kommentarzeile 1) des [Transaktionsbeispiels](#) in dieser Anleitung verwendet wurde.

Wenn Sie mit den `query()` Methoden arbeiten, wird eine Bedingung als ausgedrückt [QueryConditional](#). Die `QueryConditional` Klasse verfügt über mehrere statische praktische Methoden, mit deren Hilfe Sie die Kriterien schreiben können, die bestimmen, welche Elemente aus DynamoDB gelesen werden sollen.

Beispiele `QueryConditionals` dafür finden Sie im ersten Codebeispiel des [the section called "Query -Methodenbeispiele"](#) Abschnitts dieses Handbuchs.

## Filterausdrücke

Filterausdrücke werden bei Scan- und Abfragevorgängen verwendet, um die zurückgegebenen Elemente zu filtern.

Ein Filterausdruck wird angewendet, nachdem alle Daten aus der Datenbank gelesen wurden, sodass die Lesekosten dieselben sind, als ob es keinen Filter gäbe. Im Amazon DynamoDB Developer Guide finden Sie weitere Informationen zur Verwendung von Filterausdrücken für [Abfrage](#) - und [Scanvorgänge](#).

Das folgende Beispiel zeigt einen Filterausdruck, der einer Scananforderung hinzugefügt wurde. Die Kriterien beschränken die Anzahl der zurückgesendeten Artikel auf Artikel mit einem Preis zwischen 8,00 und einschließlich 80,00€.

```
Map<String, AttributeValue> expressionValues = Map.of(
    ":min_value", numberValue(8.00),
    ":max_value", numberValue(80.00));

ScanEnhancedRequest request = ScanEnhancedRequest.builder()
    .consistentRead(true)
    // 1. the 'attributesToProject()' method allows you to specify which
    values you want returned.
    .attributesToProject("id", "title", "authors", "price")
```



```
// 2. Filter expression limits the items returned that match the
provided criteria.
    .filterExpression(Expression.builder()
        .expression("price >= :min_value AND price <= :max_value")
        .expressionValues(expressionValues)
        .build())
    .build();
```

## Aktualisierungsausdrücke

Die Methode des DynamoDB Enhanced Client bietet eine `updateItem()` Standardmethode zum Aktualisieren von Elementen in DynamoDB. Wenn Sie jedoch mehr Funktionen benötigen, [UpdateExpressions](#) stellen Sie eine typsichere Darstellung der Syntax von [DynamoDB-Aktualisierungsausdrücken](#) bereit. Sie können es beispielsweise verwenden, um Werte `UpdateExpressions` zu erhöhen, ohne zuerst Elemente aus DynamoDB zu lesen, oder um einzelne Mitglieder zu einer Liste hinzuzufügen. Aktualisierungsausdrücke sind derzeit in benutzerdefinierten Erweiterungen für die `updateItem()` Methode verfügbar.

Ein Beispiel, das Aktualisierungsausdrücke verwendet, finden Sie im [Beispiel für eine benutzerdefinierte Erweiterung](#) in diesem Handbuch.

Weitere Informationen zu Aktualisierungsausdrücken finden Sie im [Amazon DynamoDB Developer Guide](#).

## Arbeiten mit paginierten Ergebnissen: Scans und Abfragen

Die `batch` Methoden `scan`, `query` und der DynamoDB Enhanced Client API geben Antworten mit einer oder mehreren Seiten zurück. Eine Seite enthält ein oder mehrere Elemente. Ihr Code kann die Antwort pro Seite oder einzelne Elemente verarbeiten.

Eine vom synchronen `DynamoDbEnhancedClient` Client zurückgegebene paginierte Antwort gibt ein [PagelIterable](#) Objekt zurück, während eine vom asynchronen Client zurückgegebene Antwort ein [PagePublisher](#) Objekt `DynamoDbEnhancedAsyncClient` zurückgibt.

Dieser Abschnitt untersucht die Verarbeitung paginierter Ergebnisse und enthält Beispiele, die die `Scan`- und `Abfrage`-APIs verwenden.

## Scannen einer Tabelle

Die `-scan` Methode des SDK entspricht der gleichnamigen [DynamoDB-Operation](#). Die DynamoDB Enhanced Client API bietet dieselben Optionen, verwendet jedoch ein vertrautes Objektmodell und übernimmt die Paginierung für Sie.

Zuerst untersuchen wir die PageIterable Schnittstelle, indem wir uns die scan Methode der synchronen Zuordnungsklasse ansehen [DynamoDbTable](#).

## Verwenden der synchronen API

Das folgende Beispiel zeigt die -scanMethode, die einen [Ausdruck](#) verwendet, um die zurückgegebenen Elemente zu filtern. ist [ProductCatalog](#) das Modellobjekt, das zuvor gezeigt wurde.

Der nach Kommentarzeile 1 angezeigte Filterausdruck beschränkt die ProductCatalog Elemente, die auf diejenigen mit einem Preiswert zwischen 8,00 und 80,00 zurückgegeben werden.

In diesem Beispiel werden die isbn Werte auch mit der attributesToProject Methode ausgeschlossen, die nach der Kommentarzeile 2 angezeigt wird.

In Kommentarzeile 3 pagedResult wird das PageIterable Objekt von der scan Methode zurückgegeben. Die -streamMethode gibt ein [-java.util.Stream](#) Objekt PageIterable zurück, mit dem Sie die Seiten verarbeiten können. In diesem Beispiel wird die Anzahl der Seiten gezählt und protokolliert.

Beginnend mit Kommentarzeile 4 zeigt das Beispiel zwei Varianten des Zugriffs auf die ProductCatalog Elemente. Die Version nach Kommentarzeile 2a streamt jede Seite und sortiert und protokolliert die Elemente auf jeder Seite. Die Version nach Kommentarzeile 2b überspringt die Seiteniteration und greift direkt auf die Elemente zu.

Die PageIterable Schnittstelle bietet aufgrund ihrer beiden übergeordneten Schnittstellen – und [-java.lang.Iterable](#) mehrere Möglichkeiten, Ergebnisse zu verarbeiten [SdkIterable](#). Iterable bietet die spliterator Methoden forEach, iterator und und SdkIterable die stream Methode .

```
public static void scanSync(DynamoDbTable<ProductCatalog> productCatalog) {

    Map<String, AttributeValue> expressionValues = Map.of(
        ":min_value", numberValue(8.00),
        ":max_value", numberValue(80.00));

    ScanEnhancedRequest request = ScanEnhancedRequest.builder()
        .consistentRead(true)
        // 1. the 'attributesToProject()' method allows you to specify which
        values you want returned.
        .attributesToProject("id", "title", "authors", "price")
```

```

        // 2. Filter expression limits the items returned that match the
        provided criteria.
        .filterExpression(Expression.builder()
            .expression("price >= :min_value AND price <= :max_value")
            .expressionValues(expressionValues)
            .build())
        .build();

    // 3. A PageIterable object is returned by the scan method.
    PageIterable<ProductCatalog> pagedResults = productCatalog.scan(request);
    logger.info("page count: {}", pagedResults.stream().count());

    // 4. Log the returned ProductCatalog items using two variations.
    // 4a. This version sorts and logs the items of each page.
    pagedResults.stream().forEach(p -> p.items().stream()
        .sorted(Comparator.comparing(ProductCatalog::price))
        .forEach(
            item -> logger.info(item.toString())
        ));
    // 4b. This version sorts and logs all items for all pages.
    pagedResults.items().stream()
        .sorted(Comparator.comparing(ProductCatalog::price))
        .forEach(
            item -> logger.info(item.toString())
        );
}

```

## Verwenden der asynchronen API

Die asynchrone `scan` Methode gibt Ergebnisse als `-PagePublisher` Objekt zurück. Die `PagePublisher` Schnittstelle verfügt über zwei `subscribe` Methoden, mit denen Sie Antwortseiten verarbeiten können. Eine `subscribe` Methode stammt von der `org.reactivestreams.Publisher` übergeordneten Schnittstelle. Um Seiten mit dieser ersten Option zu verarbeiten, übergeben Sie eine [Subscriber](#) Instance an die `-subscribe` Methode. Das erste folgende Beispiel zeigt die Verwendung der `-subscribe` Methode.

Die zweite `subscribe` Methode stammt von der `-SdkPublisher` Schnittstelle. Diese Version von `subscribe` akzeptiert eine [Consumer](#) anstelle einer `Subscriber`. Diese `subscribe` Methodenvariante wird im folgenden zweiten Beispiel gezeigt.

Das folgende Beispiel zeigt die asynchrone Version der `scan` Methode, die denselben Filterausdruck wie im vorherigen Beispiel verwendet.

Nach Kommentarzeile 3 gibt ein `PagePublisher` Objekt `DynamoDbAsyncTable.scan` zurück. In der nächsten Zeile erstellt der Code eine Instance der `org.reactivestreams.Subscriber` Schnittstelle, `ProductCatalogSubscriber`, die die `PagePublisher` nach Kommentarzeile 4 abonniert.

Das `-Subscriber` Objekt sammelt die `ProductCatalog` Elemente von jeder Seite in der `onNext` Methode nach Kommentarzeile 8 im `ProductCatalogSubscriber` Klassenbeispiel. Die Elemente werden in der privaten `List` Variable gespeichert und im Aufrufcode mit der `ProductCatalogSubscriber.getSubscribedItems()` Methode aufgerufen. Dies wird nach der Kommentarzeile 5 aufgerufen.

Nachdem die Liste abgerufen wurde, sortiert der Code alle `ProductCatalog` Elemente nach Preis und protokolliert jedes Element.

Die [CountDownLatch](#) in der `ProductCatalogSubscriber` Klasse blockiert den aufrufenden Thread, bis alle Elemente der Liste hinzugefügt wurden, bevor nach Kommentarzeile 5 fortgefahren wird.

```
public static void scanAsync(DynamoDbAsyncTable productCatalog) {
    ScanEnhancedRequest request = ScanEnhancedRequest.builder()
        .consistentRead(true)
        .attributesToProject("id", "title", "authors", "price")
        .filterExpression(Expression.builder()
            // 1. :min_value and :max_value are placeholders for the values
provided by the map
            .expression("price >= :min_value AND price <= :max_value")
            // 2. Two values are needed for the expression and each is
supplied as a map entry.
            .expressionValues(
                Map.of( ":min_value", numberValue(8.00),
                    ":max_value", numberValue(400_000.00)))
            .build())
        .build();

    // 3. A PagePublisher object is returned by the scan method.
    PagePublisher<ProductCatalog> pagePublisher = productCatalog.scan(request);
    ProductCatalogSubscriber subscriber = new ProductCatalogSubscriber();
    // 4. Subscribe the ProductCatalogSubscriber to the PagePublisher.
    pagePublisher.subscribe(subscriber);
    // 5. Retrieve all collected ProductCatalog items accumulated by the
subscriber.
    subscriber.getSubscribedItems().stream()
```

```

        .sorted(Comparator.comparing(ProductCatalog::price))
        .forEach(item ->
            logger.info(item.toString()));
// 6. Use a Consumer to work through each page.
pagePublisher.subscribe(page -> page
    .items().stream()
    .sorted(Comparator.comparing(ProductCatalog::price))
    .forEach(item ->
        logger.info(item.toString())))
    .join(); // If needed, blocks the subscribe() method thread until it is
finished processing.
// 7. Use a Consumer to work through each ProductCatalog item.
pagePublisher.items()
    .subscribe(product -> logger.info(product.toString()))
    .exceptionally(failure -> {
        logger.error("ERROR - ", failure);
        return null;
    })
    .join(); // If needed, blocks the subscribe() method thread until it is
finished processing.
}

```

```

private static class ProductCatalogSubscriber implements
Subscriber<Page<ProductCatalog>> {
    private CountdownLatch latch = new CountdownLatch(1);
    private Subscription subscription;
    private List<ProductCatalog> itemsFromAllPages = new ArrayList<>();

    @Override
    public void onSubscribe(Subscription sub) {
        subscription = sub;
        subscription.request(1L);
        try {
            latch.await(); // Called by main thread blocking it until latch is
released.
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    public void onNext(Page<ProductCatalog> productCatalogPage) {

```

```

        // 8. Collect all the ProductCatalog instances in the page, then ask the
        publisher for one more page.
        itemsFromAllPages.addAll(productCatalogPage.items());
        subscription.request(1L);
    }

    @Override
    public void onError(Throwable throwable) {
    }

    @Override
    public void onComplete() {
        latch.countDown(); // Call by subscription thread; latch releases.
    }

    List<ProductCatalog> getSubscribedItems() {
        return this.itemsFromAllPages;
    }
}

```

Im folgenden Beispiel wird die Version der `PagePublisher.subscribe` Methode verwendet, die eine `Consumer` Nachkommatarzeile 6 akzeptiert. Der Java-Lambda-Parameter verbraucht Seiten, die jedes Element weiter verarbeiten. In diesem Beispiel wird jede Seite verarbeitet und die Elemente auf jeder Seite werden sortiert und dann protokolliert.

```

// 6. Use a Consumer to work through each page.
pagePublisher.subscribe(page -> page
    .items().stream()
    .sorted(Comparator.comparing(ProductCatalog::price))
    .forEach(item ->
        logger.info(item.toString())))
    .join(); // If needed, blocks the subscribe() method thread until it is
finished processing.

```

Die `items` Methode zum `PagePublisher` Entpacken der Modell-Instances, sodass Ihr Code die Elemente direkt verarbeiten kann. Dieser Ansatz wird im folgenden Ausschnitt gezeigt.

```

// 7. Use a Consumer to work through each ProductCatalog item.
pagePublisher.items()
    .subscribe(product -> logger.info(product.toString()))
    .exceptionally(failure -> {
        logger.error("ERROR - ", failure);
    });

```

```
        return null;
    })
    .join(); // If needed, blocks the subscribe() method thread until it is
finished processing.
```

## Abfragen einer Tabelle

Die `-query()` Methode der `-DynamoDbTable` Klasse findet Elemente basierend auf Primärschlüsselwerten. Die `@DynamoDbPartitionKey` Anmerkung und die optionale `@DynamoDbSortKey` Anmerkung werden verwendet, um den Primärschlüssel in Ihrer Datenklasse zu definieren.

Die `query()` Methode erfordert einen Partitionsschlüsselwert, der Elemente findet, die dem angegebenen Wert entsprechen. Wenn Ihre Tabelle auch einen Sortierschlüssel definiert, können Sie Ihrer Abfrage als zusätzliche Vergleichsbedingung einen Wert hinzufügen, um die Ergebnisse zu optimieren.

Mit Ausnahme der Verarbeitung der Ergebnisse `query()` funktionieren die synchronen und asynchronen Versionen von gleich. Wie bei der `-scan` API gibt die `query` API einen `PageIterable` für einen synchronen Aufruf und einen `PagePublisher` für einen asynchronen Aufruf zurück. Wir haben die Verwendung von `PageIterable` und `PagePublisher` zuvor im `Scan`-Abschnitt besprochen.

## Query -Methodenbeispiele

Das folgende `query()` Methodencodebeispiel verwendet die `-MovieActor` Klasse. Die Datenklasse definiert einen zusammengesetzten Primärschlüssel, der aus dem `movie` Attribut für den Partitionsschlüssel und dem `actor` Attribut für den Sortierschlüssel besteht.

Die Klasse signalisiert auch, dass sie einen globalen sekundären Index namens `acting_award_year`. Der zusammengesetzte Primärschlüssel des Index besteht aus dem `actingaward` Attribut für den Partitionsschlüssel und dem `actingyear` für den Sortierschlüssel. Wenn wir später in diesem Thema zeigen, wie Indizes erstellt und verwendet werden, beziehen wir uns auf den `acting_award_year` Index.

## MovieActor-Klasse

```
package org.example.tests.model;

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbAttribute;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbBean;
```

```
import
  software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import
  software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;
import
  software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondarySortKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.util.Objects;

@DynamoDbBean
public class MovieActor implements Comparable<MovieActor> {

    private String movieName;
    private String actorName;
    private String actingAward;
    private Integer actingYear;
    private String actingSchoolName;

    @DynamoDbPartitionKey
    @DynamoDbAttribute("movie")
    public String getMovieName() {
        return movieName;
    }

    public void setMovieName(String movieName) {
        this.movieName = movieName;
    }

    @DynamoDbSortKey
    @DynamoDbAttribute("actor")
    public String getActorName() {
        return actorName;
    }

    public void setActorName(String actorName) {
        this.actorName = actorName;
    }

    @DynamoDbSecondaryPartitionKey(indexNames = "acting_award_year")
    @DynamoDbAttribute("actingaward")
    public String getActingAward() {
        return actingAward;
    }
}
```



```
public void setActingAward(String actingAward) {
    this.actingAward = actingAward;
}

@DynamoDbSecondarySortKey(indexNames = {"acting_award_year", "movie_year"})
@DynamoDbAttribute("actingyear")
public Integer getActingYear() {
    return actingYear;
}

public void setActingYear(Integer actingYear) {
    this.actingYear = actingYear;
}

@DynamoDbAttribute("actingschoolname")
public String getActingSchoolName() {
    return actingSchoolName;
}

public void setActingSchoolName(String actingSchoolName) {
    this.actingSchoolName = actingSchoolName;
}

@Override
public String toString() {
    final StringBuffer sb = new StringBuffer("MovieActor{");
    sb.append("movieName=").append(movieName).append('\ ');
    sb.append(", actorName=").append(actorName).append('\ ');
    sb.append(", actingAward=").append(actingAward).append('\ ');
    sb.append(", actingYear=").append(actingYear);
    sb.append(", actingSchoolName=").append(actingSchoolName).append('\ ');
    sb.append('}');
    return sb.toString();
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    MovieActor that = (MovieActor) o;
    return Objects.equals(movieName, that.movieName) && Objects.equals(actorName,
that.actorName) && Objects.equals(actingAward, that.actingAward) &&
```

```

Objects.equals(actingYear, that.actingYear) && Objects.equals(actingSchoolName,
that.actingSchoolName);
    }

    @Override
    public int hashCode() {
        return Objects.hash(movieName, actorName, actingAward, actingYear,
actingSchoolName);
    }

    @Override
    public int compareTo(MovieActor o) {
        if (this.movieName.compareTo(o.movieName) != 0){
            return this.movieName.compareTo(o.movieName);
        } else {
            return this.actorName.compareTo(o.actorName);
        }
    }
}
}

```

Die Codebeispiele, die auf eine Abfrage für die folgenden Elemente folgen.

### Elemente in der **MovieActor** Tabelle

```

MovieActor{movieName='movie01', actorName='actor0', actingAward='actingaward0',
actingYear=2001, actingSchoolName='null'}
MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',
actingYear=2001, actingSchoolName='actingschool1'}
MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
actingYear=2001, actingSchoolName='actingschool2'}
MovieActor{movieName='movie01', actorName='actor3', actingAward='actingaward3',
actingYear=2001, actingSchoolName='null'}
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
actingYear=2001, actingSchoolName='actingschool4'}
MovieActor{movieName='movie02', actorName='actor0', actingAward='actingaward0',
actingYear=2002, actingSchoolName='null'}
MovieActor{movieName='movie02', actorName='actor1', actingAward='actingaward1',
actingYear=2002, actingSchoolName='actingschool1'}
MovieActor{movieName='movie02', actorName='actor2', actingAward='actingaward2',
actingYear=2002, actingSchoolName='actingschool2'}
MovieActor{movieName='movie02', actorName='actor3', actingAward='actingaward3',
actingYear=2002, actingSchoolName='null'}
MovieActor{movieName='movie02', actorName='actor4', actingAward='actingaward4',
actingYear=2002, actingSchoolName='actingschool4'}

```

```

MovieActor{movieName='movie03', actorName='actor0', actingAward='actingaward0',
  actingYear=2003, actingSchoolName='null'}
MovieActor{movieName='movie03', actorName='actor1', actingAward='actingaward1',
  actingYear=2003, actingSchoolName='actingschool1'}
MovieActor{movieName='movie03', actorName='actor2', actingAward='actingaward2',
  actingYear=2003, actingSchoolName='actingschool2'}
MovieActor{movieName='movie03', actorName='actor3', actingAward='actingaward3',
  actingYear=2003, actingSchoolName='null'}
MovieActor{movieName='movie03', actorName='actor4', actingAward='actingaward4',
  actingYear=2003, actingSchoolName='actingschool4'}

```

Der folgende Code definiert zwei [QueryConditional](#) Instances. QueryConditionals arbeiten mit Schlüsselwerten – entweder nur der Partitionsschlüssel oder in Kombination mit dem Sortierschlüssel – und entsprechen den [bedingten Schlüsselausdrücken](#) der DynamoDB-Service-API. Nach der Kommentarzeile 1 definiert das Beispiel die `keyEqual` Instance, die Elemente mit dem Partitionswert abgleicht **movie01**.

In diesem Beispiel wird auch ein Filterausdruck definiert, der jedes Element filtert, für das nach der Kommentarzeile 2 kein **actingschoolname** steht.

Nach Kommentarzeile 3 zeigt das Beispiel die [QueryEnhancedRequest](#) Instance, die der Code an die `DynamoDbTable.query()` Methode übergibt. Dieses Objekt kombiniert die Schlüsselbedingung und den Filter, die das SDK verwendet, um die Anforderung an den DynamoDB-Service zu generieren.

```

public static void query(DynamoDbTable movieActorTable) {

    // 1. Define a QueryConditional instance to return items matching a partition
    value.
    QueryConditional keyEqual = QueryConditional.keyEqualTo(b ->
b.partitionValue("movie01"));
    // 1a. Define a QueryConditional that adds a sort key criteria to the partition
    value criteria.
    QueryConditional sortGreaterThanOrEqualTo =
QueryConditional.sortGreaterThanOrEqualTo(b ->
b.partitionValue("movie01").sortValue("actor2"));
    // 2. Define a filter expression that filters out items whose attribute value
    is null.
    final Expression filterOutNoActingschoolname =
Expression.builder().expression("attribute_exists(actingschoolname)").build();

    // 3. Build the query request.

```

```

QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder()
    .queryConditional(keyEqual)
    .filterExpression(filterOutNoActingschoolname)
    .build();
// 4. Perform the query.
PageIterable<MovieActor> pagedResults = movieActorTable.query(tableQuery);
logger.info("page count: {}", pagedResults.stream().count()); // Log number of
pages.

pagedResults.items().stream()
    .sorted()
    .forEach(
        item -> logger.info(item.toString()) // Log the sorted list of
items.
    );

```

Im Folgenden finden Sie die Ausgabe der Ausführung der -Methode. Die Ausgabe zeigt Elemente mit dem `movieName` Wert `movie01` und keine Elemente mit `actingSchoolName` gleich **null**.

```

2023-03-05 13:11:05 [main] INFO org.example.tests.QueryDemo:46 - page count: 1
2023-03-05 13:11:05 [main] INFO org.example.tests.QueryDemo:51 -
MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',
actingYear=2001, actingSchoolName='actingschool1'}
2023-03-05 13:11:05 [main] INFO org.example.tests.QueryDemo:51 -
MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
actingYear=2001, actingSchoolName='actingschool2'}
2023-03-05 13:11:05 [main] INFO org.example.tests.QueryDemo:51 -
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
actingYear=2001, actingSchoolName='actingschool4'}

```

In der folgenden Variation der Abfrageanforderung, die zuvor nach Kommentarzeile 3 gezeigt wurde, ersetzt der Code die durch `keyEqual` `QueryConditional` die `sortGreaterThanOrEqualToQueryConditional`, die nach Kommentarzeile 1a definiert wurde. Der folgende Code entfernt auch den Filterausdruck.

```

QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder()
    .queryConditional(sortGreaterThanOrEqualTo)

```

Da diese Tabelle über einen zusammengesetzten Primärschlüssel verfügt, benötigen alle `QueryConditional` Instances einen Partitionsschlüsselwert. -`QueryConditional` Methoden, die

mit `beginnen, sort...` geben an, dass ein Sortierschlüssel erforderlich ist. Die Ergebnisse sind nicht sortiert.

Die folgende Ausgabe zeigt die Ergebnisse der Abfrage an. Die Abfrage gibt Elemente zurück, die einen `movieName` Wert haben, der gleich `movie01` ist, und nur Elemente, die einen `actorName` Wert haben, der größer oder gleich `Akteur2` ist. Da der Filter entfernt wurde, gibt die Abfrage Elemente zurück, die keinen Wert für das `actingSchoolName` Attribut haben.

```
2023-03-05 13:15:00 [main] INFO org.example.tests.QueryDemo:46 - page count: 1
2023-03-05 13:15:00 [main] INFO org.example.tests.QueryDemo:51 -
  MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
  actingYear=2001, actingSchoolName='actingschool2'}
2023-03-05 13:15:00 [main] INFO org.example.tests.QueryDemo:51 -
  MovieActor{movieName='movie01', actorName='actor3', actingAward='actingaward3',
  actingYear=2001, actingSchoolName='null'}
2023-03-05 13:15:00 [main] INFO org.example.tests.QueryDemo:51 -
  MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
  actingYear=2001, actingSchoolName='actingschool4'}
```

## Ausführen von Batchoperationen

Die DynamoDB Enhanced Client API bietet zwei Batch-Methoden, [batchGetItem\(\)](#) und [batchWriteItem\(\)](#).

### **batchGetItem()** Beispiel für

Mit der [DynamoDbTable.batchGetItem\(\)](#) Methode können Sie bis zu 100 einzelne Elemente in mehreren Tabellen in einer Gesamtanforderung abrufen. Im folgenden Beispiel werden die zuvor gezeigten [MovieActor](#) Datenklassen [Customer](#) und verwendet.

Im Beispiel nach den Zeilen 1 und 2 erstellen Sie [ReadBatch](#) Objekte, die Sie später nach der Kommentarzeile 3 als Parameter zur `batchGetItem()` Methode hinzufügen. Der Code nach Kommentarzeile 1 erstellt den Batch zum Lesen aus der `Customer` Tabelle. Der Code nach Kommentarzeile 1a zeigt die Verwendung eines [GetItemEnhancedRequest](#) Builders, der Primärschlüsselwerte verwendet, um das zu lesende Element anzugeben. Im Gegensatz zur Angabe von Schlüsselwerten zum Anfordern eines Elements können Sie eine Datenklasse verwenden, um ein Element anzufordern, wie nach der Kommentarzeile 1b gezeigt. Das SDK extrahiert die Schlüsselwerte hinter den Kulissen, bevor die Anforderung gesendet wird.

Wenn Sie das Element mit dem schlüsselbasierten Ansatz angeben, wie in den beiden Anweisungen nach 2a gezeigt, können Sie auch angeben, dass DynamoDB einen [Strongly-Consistent-](#)

[Lesevorgang](#) durchführen soll. Wenn die `-consistentRead()` Methode verwendet wird, muss sie für alle angeforderten Elemente für dieselbe Tabelle verwendet werden.

Um die Elemente abzurufen, die DynamoDB gefunden hat, verwenden Sie die [-resultsForTable\(\)](#) Methode, die nach der Kommentarzeile 4 angezeigt wird. Rufen Sie die Methode für jede Tabelle auf, die in der Anforderung gelesen wurde. `resultsForTable()` gibt eine Liste der gefundenen Elemente zurück, die Sie mit einer beliebigen `java.util.List` Methode verarbeiten können. In diesem Beispiel wird jedes Element protokolliert.

Um Elemente zu ermitteln, die DynamoDB nicht verarbeitet hat, verwenden Sie den Ansatz nach Kommentarzeile 5. Die `BatchGetResultPage` Klasse verfügt über die [-unprocessedKeysForTable\(\)](#) Methode, die Ihnen Zugriff auf jeden unverarbeiteten Schlüssel gewährt. Die [BatchGetItem API-Referenz](#) enthält weitere Informationen zu Situationen, die zu unverarbeiteten Elementen führen.

```
public static void batchGetItemExample(DynamoDbEnhancedClient enhancedClient,
                                       DynamoDbTable<Customer> customerTable,
                                       DynamoDbTable<MovieActor> movieActorTable) {

    Customer customer2 = new Customer();
    customer2.setId("2");
    customer2.setEmail("cust2@example.org");

    // 1. Build a batch to read from the Customer table.
    ReadBatch customerBatch = ReadBatch.builder(Customer.class)
        .mappedTableResource(customerTable)
        // 1a. Specify the primary key values for the item.
        .addGetItem(b -> b.key(k ->
k.partitionValue("1").sortValue("cust1@orgname.org")))
        // 1b. Alternatively, supply a data class instances to provide the
primary key values.
        .addGetItem(customer2)
        .build();

    // 2. Build a batch to read from the MovieActor table.
    ReadBatch moveActorBatch = ReadBatch.builder(MovieActor.class)
        .mappedTableResource(movieActorTable)
        // 2a. Call consistentRead(Boolean.TRUE) for each item for the same
table.
        .addGetItem(b -> b.key(k ->
k.partitionValue("movie01").sortValue("actor1")).consistentRead(Boolean.TRUE))
```

```

        .addGetItem(b -> b.key(k ->
k.partitionValue("movie01").sortValue("actor4")).consistentRead(Boolean.TRUE))
        .build());

// 3. Add ReadBatch objects to the request.
BatchGetResultPageIterable resultPages = enhancedClient.batchGetItem(b ->
b.readBatches(customerBatch, moveActorBatch));

// 4. Retrieve the successfully requested items from each table.
resultPages.resultsForTable(customerTable).forEach(item ->
logger.info(item.toString()));
resultPages.resultsForTable(movieActorTable).forEach(item ->
logger.info(item.toString()));

// 5. Retrieve the keys of the items requested but not processed by the
service.
resultPages.forEach((BatchGetResultPage pageResult) -> {
    pageResult.unprocessedKeysForTable(customerTable).forEach(key ->
logger.info("Unprocessed item key: " + key.toString()));
    pageResult.unprocessedKeysForTable(movieActorTable).forEach(key ->
logger.info("Unprocessed item key: " + key.toString()));
});
}

```

Angenommen, die folgenden Elemente befinden sich in den beiden Tabellen, bevor Sie den Beispielcode ausführen.

## Elemente in Tabellen

```

Customer [id=1, name=CustName1, email=cust1@example.org,
regDate=2023-03-31T15:46:27.688Z]
Customer [id=2, name=CustName2, email=cust2@example.org,
regDate=2023-03-31T15:46:28.688Z]
Customer [id=3, name=CustName3, email=cust3@example.org,
regDate=2023-03-31T15:46:29.688Z]
Customer [id=4, name=CustName4, email=cust4@example.org,
regDate=2023-03-31T15:46:30.688Z]
Customer [id=5, name=CustName5, email=cust5@example.org,
regDate=2023-03-31T15:46:31.689Z]
MovieActor{movieName='movie01', actorName='actor0', actingAward='actingaward0',
actingYear=2001, actingSchoolName='null'}
MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',
actingYear=2001, actingSchoolName='actingschool1'}

```

```

MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
  actingYear=2001, actingSchoolName='actingschool2'}
MovieActor{movieName='movie01', actorName='actor3', actingAward='actingaward3',
  actingYear=2001, actingSchoolName='null'}
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
  actingYear=2001, actingSchoolName='actingschool4'}

```

Die folgende Ausgabe zeigt die Elemente, die nach der Kommentarzeile 4 zurückgegeben und protokolliert wurden.

```

Customer [id=1, name=CustName1, email=cust1@example.org,
  regDate=2023-03-31T15:46:27.688Z]
Customer [id=2, name=CustName2, email=cust2@example.org,
  regDate=2023-03-31T15:46:28.688Z]
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
  actingYear=2001, actingSchoolName='actingschool4'}
MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',
  actingYear=2001, actingSchoolName='actingschool1'}

```

### **batchWriteItem()** Beispiel für

Die `-batchWriteItem()` Methode platziert oder löscht mehrere Elemente in einer oder mehreren Tabellen. Sie können bis zu 25 einzelne Put- oder Löschvorgänge in der Anforderung angeben. Im folgenden Beispiel werden die zuvor gezeigten [MovieActor](#) Modellklassen [ProductCatalog](#) und verwendet.

`WriteBatch` -Objekte werden nach den Kommentarzeilen 1 und 2 erstellt. Für die `ProductCatalog` Tabelle fügt der Code ein Element ein und löscht ein Element. Für die `MovieActor` Tabelle nach Kommentarzeile 2 fügt der Code zwei Elemente ein und löscht eines.

Die `batchWriteItem` Methode wird nach der Kommentarzeile 3 aufgerufen. Der [builder](#) Parameter stellt die Batch-Anforderungen für jede Tabelle bereit.

Das zurückgegebene [BatchWriteResult](#) Objekt bietet separate Methoden für jede Operation zum Anzeigen unverarbeiteter Anforderungen. Der Code nach Kommentarzeile 4a stellt die Schlüssel für unverarbeitete Löschanforderungen bereit und der Code nach Kommentarzeile 4b stellt die unverarbeiteten Put-Elemente bereit.

```

    public static void batchWriteItemExample(DynamoDbEnhancedClient enhancedClient,
                                             DynamoDbTable<ProductCatalog>
catalogTable,

```



```

DynamoDbTable<MovieActor> movieActorTable)
{
    // 1. Build a batch to write to the ProductCatalog table.
    WriteBatch products = WriteBatch.builder(ProductCatalog.class)
        .mappedTableResource(catalogTable)
        .addPutItem(b -> b.item(getProductCatItem1()))
        .addDeleteItem(b -> b.key(k -> k
            .partitionValue(getProductCatItem2().id())
            .sortValue(getProductCatItem2().title()))
        .build();

    // 2. Build a batch to write to the MovieActor table.
    WriteBatch movies = WriteBatch.builder(MovieActor.class)
        .mappedTableResource(movieActorTable)
        .addPutItem(getMovieActorYeoh())
        .addPutItem(getMovieActorBlanchettPartial())
        .addDeleteItem(b -> b.key(k -> k
            .partitionValue(getMovieActorStreep().getMovieName())
            .sortValue(getMovieActorStreep().getActorName()))
        .build();

    // 3. Add WriteBatch objects to the request.
    BatchWriteResult batchWriteResult = enhancedClient.batchWriteItem(b ->
b.writeBatches(products, movies));
    // 4. Retrieve keys for items the service did not process.
    // 4a. 'unprocessedDeleteItemsForTable()' returns keys for delete requests that
did not process.
    if (batchWriteResult.unprocessedDeleteItemsForTable(movieActorTable).size() >
0) {
        batchWriteResult.unprocessedDeleteItemsForTable(movieActorTable).forEach(key ->
            logger.info(key.toString()));
    }
    // 4b. 'unprocessedPutItemsForTable()' returns keys for put requests that did
not process.
    if (batchWriteResult.unprocessedPutItemsForTable(catalogTable).size() > 0) {
        batchWriteResult.unprocessedPutItemsForTable(catalogTable).forEach(key ->
            logger.info(key.toString()));
    }
}

```

Die folgenden Hilfsmethoden stellen die Modellobjekte für die Put- und Delete-Operationen bereit.

## Hilfsmethoden

```
public static ProductCatalog getProductCatItem1() {
    return ProductCatalog.builder()
        .id(2)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))
        .price(BigDecimal.valueOf(30.22))
        .title("Title 55")
        .build();
}

public static ProductCatalog getProductCatItem2() {
    return ProductCatalog.builder()
        .id(4)
        .price(BigDecimal.valueOf(40.00))
        .title("Title 1")
        .build();
}

public static MovieActor getMovieActorBlanchettPartial() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Cate Blanchett");
    movieActor.setMovieName("Blue Jasmine");
    movieActor.setActingYear(2023);
    movieActor.setActingAward("Best Actress");
    return movieActor;
}

public static MovieActor getMovieActorStreep() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Meryl Streep");
    movieActor.setMovieName("Sophie's Choice");
    movieActor.setActingYear(1982);
    movieActor.setActingAward("Best Actress");
    movieActor.setActingSchoolName("Yale School of Drama");
    return movieActor;
}

public static MovieActor getMovieActorYeoh(){
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Michelle Yeoh");
    movieActor.setMovieName("Everything Everywhere All at Once");
    movieActor.setActingYear(2023);
}
```

```
movieActor.setActingAward("Best Actress");
movieActor.setActingSchoolName("Royal Academy of Dance");
return movieActor;
}
```

Angenommen, die Tabellen enthalten die folgenden Elemente, bevor Sie den Beispielcode ausführen.

```
MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2013, actingSchoolName='National Institute of Dramatic Art'}
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
```

Nachdem der Beispielcode abgeschlossen ist, enthalten die Tabellen die folgenden Elemente.

```
MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2013, actingSchoolName='null'}
MovieActor{movieName='Everything Everywhere All at Once', actorName='Michelle Yeoh', actingAward='Best Actress', actingYear=2023, actingSchoolName='Royal Academy of Dance'}
ProductCatalog{id=2, title='Title 55', isbn='1-565-85698', authors=[a, b], price=30.22}
```

Beachten Sie in der `MovieActor` Tabelle, dass das `Blue Jasmine` Filmelement durch das Element ersetzt wurde, das in der `Put`-Anforderung verwendet wurde, die über die `getMovieActorBlanchettPartial()` Hilfsmethode abgerufen wurde. Wenn kein Wert für ein Daten-Bean-Attribut angegeben wurde, wird der Wert in der Datenbank entfernt. Aus diesem Grund ist `actingSchoolName` das Ergebnis für das `Blue Jasmine` Filmelement null.

#### Note

Obwohl die API-Dokumentation darauf hindeutet, dass Bedingungsausdrücke verwendet werden können und dass verbrauchte Kapazitäts- und Sammlungsmetriken mit einzelnen [Put](#)- und [Löschenforderungen](#) zurückgegeben werden können, ist dies in einem Batch-Write-Szenario nicht der Fall. Um die Leistung für Batchoperationen zu verbessern, werden diese einzelnen Optionen ignoriert.

## Ausführen von Transaktionsoperationen

Die DynamoDB Enhanced Client API stellt die `transactWriteItems()` Methoden `transactGetItems()` und bereit. Die Transaktionsmethoden des SDK for Java bieten Atomizität, Konsistenz, Isolation und Haltbarkeit (ACID) in DynamoDB-Tabellen, sodass Sie die Richtigkeit der Daten in Ihren Anwendungen wahren können.

### `transactGetItems()` Beispiel für

Die [transactGetItems\(\)](#) Methode akzeptiert bis zu 100 einzelne Anforderungen für Elemente. Alle Elemente werden in einer einzigen atomaren Transaktion gelesen. Das Amazon-DynamoDB-Entwicklerhandbuch enthält Informationen über die [Bedingungen, die dazu führen, dass eine transactGetItems\(\) Methode fehlschlägt](#), sowie über die beim Aufrufen von verwendete Isolationsstufefür [transactGetItem\(\)](#).

Nach der Kommentarzeile 1 im folgenden Beispiel ruft der Code die `transactGetItems()` Methode mit einem [builder](#) Parameter auf. Der des Builders [addItem\(\)](#) wird dreimal mit einem Datenobjekt aufgerufen, das die Schlüsselwerte enthält, die das SDK zum Generieren der endgültigen Anforderung verwendet.

Die Anforderung gibt eine Liste von [Document](#) Objekten nach Kommentarzeile 2 zurück. Die Liste der zurückgegebenen Dokumente enthält nicht null Dokumenten-[Instances](#) von Elementdaten in derselben Reihenfolge wie angefordert. Die [Document.getItem\(MappedTableResource<T> mappedTableResource\)](#) Methode konvertiert ein nicht typisiertes Document Objekt in ein typisiertes Java-Objekt, wenn Elementdaten zurückgegeben wurden, andernfalls gibt die Methode null zurück.

```
public static void transactGetItemsExample(DynamoDbEnhancedClient enhancedClient,
                                          DynamoDbTable<ProductCatalog>
catalogTable,
                                          DynamoDbTable<MovieActor>
movieActorTable) {

    // 1. Request three items from two tables using a builder.
    final List<Document> documents = enhancedClient.transactGetItems(b -> b
        .addItem(catalogTable,
Key.builder().partitionValue(2).sortValue("Title 55").build())
        .addItem(movieActorTable, Key.builder().partitionValue("Sophie's
Choice").sortValue("Meryl Streep").build())
        .addItem(movieActorTable, Key.builder().partitionValue("Blue
Jasmine").sortValue("Cate Blanchett").build())
```

```

        .build());

// 2. A list of Document objects is returned in the same order as requested.
ProductCatalog title55 = documents.get(0).getItem(catalogTable);
if (title55 != null) {
    logger.info(title55.toString());
}

MovieActor sophiesChoice = documents.get(1).getItem(movieActorTable);
if (sophiesChoice != null) {
    logger.info(sophiesChoice.toString());
}

// 3. The getItem() method returns null if the Document object contains no item
from DynamoDB.
MovieActor blueJasmine = documents.get(2).getItem(movieActorTable);
if (blueJasmine != null) {
    logger.info(blueJasmine.toString());
}
}

```

Die DynamoDB-Tabellen enthalten die folgenden Elemente, bevor das Codebeispiel ausgeführt wird.

```

ProductCatalog{id=2, title='Title 55', isbn='orig_isbn', authors=[b, g], price=10}
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best
Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}

```

Die folgende Ausgabe wird protokolliert. Wenn ein Element angefordert, aber nicht gefunden wird, wird es nicht wie bei der Anfrage für den Film mit dem Namen zurückgegeben `Blue Jasmine`.

```

ProductCatalog{id=2, title='Title 55', isbn='orig_isbn', authors=[b, g], price=10}
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best
Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}

```

## Beispiele für `transactWriteItems()`

Der [transactWriteItems\(\)](#) akzeptiert bis zu 100 Put-, Update- oder Löschaktionen in einer einzigen atomaren Transaktion über mehrere Tabellen hinweg. Das Amazon-DynamoDB-Entwicklerhandbuch enthält Details zu Einschränkungen und Fehlerbedingungen der zugrunde liegenden [DynamoDB-Serviceoperation](#).

## Einfaches Beispiel

Im folgenden Beispiel werden vier Operationen für zwei Tabellen angefordert. Die entsprechenden Modellklassen [ProductCatalog](#) und [MovieActor](#) wurden zuvor gezeigt.

Jede der drei möglichen Operationen – Setzen, Aktualisieren und Löschen – verwendet einen dedizierten Anforderungsparameter, um die Details anzugeben.

Der Code nach Kommentarzeile 1 zeigt die einfache Variante der `addPutItem()` Methode. Die Methode akzeptiert ein [MappedTableResource](#) Objekt und die zu setzende Datenobjekt-Instance. Die Anweisung nach Kommentarzeile 2 zeigt die Variante, die eine [TransactPutItemEnhancedRequest](#) Instance akzeptiert. Mit dieser Variante können Sie der Anforderung weitere Optionen hinzufügen, z. B. einen Bedingungsausdruck. Ein nachfolgendes [Beispiel](#) zeigt einen Bedingungsausdruck für eine einzelne Operation.

Ein Aktualisierungsvorgang wird nach der Kommentarzeile 3 angefordert.

[TransactUpdateItemEnhancedRequest](#) verfügt über eine `-ignoreNulls()` Methode, mit der Sie konfigurieren können, was das SDK mit `null` Werten für das Modellobjekt macht. Wenn die `ignoreNulls()` Methode „true“ zurückgibt, entfernt das SDK die Attributwerte der Tabelle für Datenobjektattribute, die `isNull` sind. Wenn die `ignoreNulls()` Methode `false` zurückgibt, fordert das SDK den DynamoDB-Service auf, die Attribute aus dem Element in der Tabelle zu entfernen. Der Standardwert für `ignoreNulls` ist `false`.

Die Anweisung nach Kommentarzeile 4 zeigt die Variante einer Löschanforderung, die ein Datenobjekt verwendet. Der erweiterte Client extrahiert die Schlüsselwerte, bevor er die endgültige Anfrage sendet.

```
public static void transactWriteItems(DynamoDbEnhancedClient enhancedClient,
                                     DynamoDbTable<ProductCatalog> catalogTable,
                                     DynamoDbTable<MovieActor> movieActorTable) {

    enhancedClient.transactWriteItems(b -> b
        // 1. Simplest variation of put item request.
        .addPutItem(catalogTable, getProductCatId2())
        // 2. Put item request variation that accommodates condition
expressions.
        .addPutItem(movieActorTable,
TransactPutItemEnhancedRequest.builder(MovieActor.class)
            .item(getMovieActorStreep())
```

```

.conditionExpression(Expression.builder().expression("attribute_not_exists
(movie)").build())
        .build())
    // 3. Update request that does not remove attribute values on the table
    if the data object's value is null.
        .addUpdateItem(catalogTable,
TransactUpdateItemEnhancedRequest.builder(ProductCatalog.class)
        .item(getProductCatId4ForUpdate())
        .ignoreNulls(Boolean.TRUE)
        .build())
    // 4. Variation of delete request that accepts a data object. The key
    values are extracted for the request.
        .addDeleteItem(movieActorTable, getMovieActorBlanchett())
    );
}

```

Die folgenden Hilfsmethoden stellen die Datenobjekte für die add\*Item Parameter bereit.

### Hilfsmethoden

```

public static ProductCatalog getProductCatId2() {
    return ProductCatalog.builder()
        .id(2)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))
        .price(BigDecimal.valueOf(30.22))
        .title("Title 55")
        .build();
}

public static ProductCatalog getProductCatId4ForUpdate() {
    return ProductCatalog.builder()
        .id(4)
        .price(BigDecimal.valueOf(40.00))
        .title("Title 1")
        .build();
}

public static MovieActor getMovieActorBlanchett() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Cate Blanchett");
    movieActor.setMovieName("Tar");
    movieActor.setActingYear(2022);
}

```

```

        movieActor.setActingAward("Best Actress");
        movieActor.setActingSchoolName("National Institute of Dramatic Art");
        return movieActor;
    }

    public static MovieActor getMovieActorStreep() {
        MovieActor movieActor = new MovieActor();
        movieActor.setActorName("Meryl Streep");
        movieActor.setMovieName("Sophie's Choice");
        movieActor.setActingYear(1982);
        movieActor.setActingAward("Best Actress");
        movieActor.setActingSchoolName("Yale School of Drama");
        return movieActor;
    }

```

Die DynamoDB-Tabellen enthalten die folgenden Elemente, bevor das Codebeispiel ausgeführt wird.

```

1 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2 | MovieActor{movieName='Tar', actorName='Cate Blanchett', actingAward='Best Actress',
  actingYear=2022, actingSchoolName='National Institute of Dramatic Art'}

```

Die folgenden Elemente befinden sich in den Tabellen, nachdem die Ausführung des Codes abgeschlossen ist.

```

3 | ProductCatalog{id=2, title='Title 55', isbn='1-565-85698', authors=[a, b],
  price=30.22}
4 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=40.0}
5 | MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best
  Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}

```

Das Element in Zeile 2 wurde gelöscht und die Zeilen 3 und 5 zeigen die Elemente an, die eingefügt wurden. Zeile 4 zeigt das Update von Zeile 1. Der price Wert ist der einzige Wert, der sich für das Element geändert hat. Wenn „false“ zurückgegeben `ignoreNulls()` hätte, würde Zeile 4 wie die folgende Zeile aussehen.

```
ProductCatalog{id=4, title='Title 1', isbn='null', authors=null, price=40.0}
```

### Beispiel für eine Bedingungsprüfung

Das folgende Beispiel zeigt die Verwendung einer Bedingungsprüfung. Eine Bedingungsprüfung wird verwendet, um zu überprüfen, ob ein Element vorhanden ist, oder um den Zustand bestimmter



Attribute eines Elements in der Datenbank zu überprüfen. Das in der Bedingungsprüfung überprüfte Element kann nicht in einer anderen Operation in der Transaktion verwendet werden.

### Note

Sie können nicht in derselben Transaktion mit mehreren Operationen auf das gleiche Element abzielen. Sie können beispielsweise keine Bedingungsprüfung durchführen und auch versuchen, dasselbe Element in derselben Transaktion zu aktualisieren.

Das Beispiel zeigt eine jeder Art von Operation in einer Anforderung für transaktionale Schreibelemente. Nach Kommentarzeile 2 liefert die `addConditionCheck()` Methode die Bedingung, die die Transaktion nicht besteht, wenn der `conditionExpression` Parameter mit ausgewertet wird `false`. Der Bedingungsausdruck, der von der Methode zurückgegeben wird, die im Block Hilfsmethoden angezeigt Sophie's Choice wird, prüft, ob das Jahr für den Film nicht entspricht 1982. Ist dies der Fall, wird der Ausdruck zu ausgewertet `false` und die Transaktion schlägt fehl.

In diesem Handbuch werden [Ausdrücke](#) in einem anderen Thema ausführlich behandelt.

```
public static void conditionCheckFailExample(DynamoDbEnhancedClient enhancedClient,
                                             DynamoDbTable<ProductCatalog>
catalogTable,
                                             DynamoDbTable<MovieActor>
movieActorTable) {

    try {
        enhancedClient.transactWriteItems(b -> b
            // 1. Perform one of each type of operation with the next three
methods.
                .addPutItem(catalogTable,
TransactPutItemEnhancedRequest.builder(ProductCatalog.class)
                    .item(getProductCatId2()).build())
                .addUpdateItem(catalogTable,
TransactUpdateItemEnhancedRequest.builder(ProductCatalog.class)
                    .item(getProductCatId4ForUpdate())
                    .ignoreNulls(Boolean.TRUE).build())
                .addDeleteItem(movieActorTable,
TransactDeleteItemEnhancedRequest.builder()
                    .key(b1 -> b1
```

```

.partitionValue(getMovieActorBlanchett().getMovieName())

.sortValue(getMovieActorBlanchett().getActorName()).build()
    // 2. Add a condition check on a table item that is not involved in
another operation in this request.
    .addConditionCheck(movieActorTable, ConditionCheck.builder()
        .conditionExpression(buildConditionCheckExpression())
        .key(k -> k
            .partitionValue("Sophie's Choice")
            .sortValue("Meryl Streep"))
    // 3. Specify the request to return existing values from
the item if the condition evaluates to true.

.returnValuesOnConditionCheckFailure(ReturnValuesOnConditionCheckFailure.ALL_OLD)
    .build())
    .build());
// 4. Catch the exception if the transaction fails and log the information.
} catch (TransactionCanceledException ex) {
    ex.cancellationReasons().stream().forEach(cancellationReason -> {
        logger.info(cancellationReason.toString());
    });
}
}
}

```

Die folgenden Hilfsmethoden werden im vorherigen Codebeispiel verwendet.

## Hilfsmethoden

```

private static Expression buildConditionCheckExpression() {
    Map<String, AttributeValue> expressionValue = Map.of(
        ":year", numberValue(1982));

    return Expression.builder()
        .expression("actingyear <> :year")
        .expressionValues(expressionValue)
        .build();
}

public static ProductCatalog getProductCatId2() {
    return ProductCatalog.builder()
        .id(2)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))

```

```

        .price(BigDecimal.valueOf(30.22))
        .title("Title 55")
        .build();
    }

    public static ProductCatalog getProductCatId4ForUpdate() {
        return ProductCatalog.builder()
            .id(4)
            .price(BigDecimal.valueOf(40.00))
            .title("Title 1")
            .build();
    }

    public static MovieActor getMovieActorBlanchett() {
        MovieActor movieActor = new MovieActor();
        movieActor.setActorName("Cate Blanchett");
        movieActor.setMovieName("Blue Jasmine");
        movieActor.setActingYear(2013);
        movieActor.setActingAward("Best Actress");
        movieActor.setActingSchoolName("National Institute of Dramatic Art");
        return movieActor;
    }
}

```

Die DynamoDB-Tabellen enthalten die folgenden Elemente, bevor das Codebeispiel ausgeführt wird.

```

1 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2 | MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
3 | MovieActor{movieName='Tar', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2022, actingSchoolName='National Institute of Dramatic Art'}

```

Die folgenden Elemente befinden sich in den Tabellen, nachdem die Ausführung des Codes abgeschlossen ist.

```

ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
MovieActor{movieName='Tar', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2022, actingSchoolName='National Institute of Dramatic Art'}

```

Elemente bleiben in den Tabellen unverändert, da die Transaktion fehlgeschlagen ist. Der `actingYear` Wert für den Film `Sophie's Choice` ist 1982, wie in Zeile 2 der Elemente in der Tabelle gezeigt, bevor die `transactWriteItem()` Methode aufgerufen wird.

Um die Stornierungsinformationen für die Transaktion zu erfassen, schließen Sie den `transactWriteItems()` Methodenaufruf in einen `-tryBlock` und `catch` die ein [TransactionCanceledException](#). Nach Kommentarzeile 4 des Beispiels protokolliert der Code jedes [CancellationReason](#) Objekt. Da der Code nach Kommentarzeile 3 des Beispiels angibt, dass Werte für das Element zurückgegeben werden sollen, das dazu geführt hat, dass die Transaktion fehlschlägt, zeigt das Protokoll die Rohdatenbankwerte für das `Sophie's Choice` Filmelement an.

```
CancellationReason(Code=None)
CancellationReason(Code=None)
CancellationReason(Code=None)
CancellationReason(Item={actor=AttributeValue(S=Meryl Streep),
  movie=AttributeValue(S=Sophie's Choice), actingaward=AttributeValue(S=Best Actress),
  actingyear=AttributeValue(N=1982), actingschoolname=AttributeValue(S=Yale School of
  Drama)}, -
  Code=ConditionalCheckFailed, Message=The conditional request failed.)
```

### Beispiel für eine Einzeloperationsbedingung

Das folgende Beispiel zeigt die Verwendung einer Bedingung für eine einzelne Operation in einer Transaktionsanforderung. Die Löschoperation nach Kommentarzeile 1 enthält eine Bedingung, die den Wert des Zielelements der Operation anhand der Datenbank prüft. In diesem Beispiel gibt der Bedingungsausdruck, der mit der Hilfsmethode nach Kommentarzeile 2 erstellt wurde, an, dass das Element aus der Datenbank gelöscht werden soll, wenn das aktive Jahr des Films nicht gleich 2013 ist.

[Ausdrücke](#) werden weiter unten in diesem Leitfaden erörtert.

```
public static void singleOperationConditionFailExample(DynamoDbEnhancedClient
enhancedClient,

DynamoDbTable<ProductCatalog> catalogTable,

DynamoDbTable<MovieActor>
movieActorTable) {
    try {
        enhancedClient.transactWriteItems(b -> b
```

```

        .addPutItem(catalogTable,
TransactPutItemEnhancedRequest.builder(ProductCatalog.class)
            .item(getProductCatId2())
            .build())
        .addUpdateItem(catalogTable,
TransactUpdateItemEnhancedRequest.builder(ProductCatalog.class)
            .item(getProductCatId4ForUpdate())
            .ignoreNulls(Boolean.TRUE).build())
// 1. Delete operation that contains a condition expression
        .addDeleteItem(movieActorTable,
TransactDeleteItemEnhancedRequest.builder()
            .key((Key.Builder k) -> {
                MovieActor blanchett = getMovieActorBlanchett();
                k.partitionValue(blanchett.getMovieName())
                    .sortValue(blanchett.getActorName());
            })
            .conditionExpression(buildDeleteItemExpression()))

        .returnValuesOnConditionCheckFailure(ReturnValuesOnConditionCheckFailure.ALL_OLD)
            .build())
        .build());
    } catch (TransactionCanceledException ex) {
        ex.cancellationReasons().forEach(cancellationReason ->
logger.info(cancellationReason.toString()));
    }
}

// 2. Provide condition expression to check if 'actingyear' is not equal to 2013.
private static Expression buildDeleteItemExpression() {
    Map<String, AttributeValue> expressionValue = Map.of(
        ":year", numberValue(2013));

    return Expression.builder()
        .expression("actingyear <> :year")
        .expressionValues(expressionValue)
        .build();
}
}

```

Die folgenden Hilfsmethoden werden im vorherigen Codebeispiel verwendet.

## Hilfsmethoden

```

public static ProductCatalog getProductCatId2() {
    return ProductCatalog.builder()

```

```

        .id(2)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))
        .price(BigDecimal.valueOf(30.22))
        .title("Title 55")
        .build();
    }

    public static ProductCatalog getProductCatId4ForUpdate() {
        return ProductCatalog.builder()
            .id(4)
            .price(BigDecimal.valueOf(40.00))
            .title("Title 1")
            .build();
    }

    public static MovieActor getMovieActorBlanchett() {
        MovieActor movieActor = new MovieActor();
        movieActor.setActorName("Cate Blanchett");
        movieActor.setMovieName("Blue Jasmine");
        movieActor.setActingYear(2013);
        movieActor.setActingAward("Best Actress");
        movieActor.setActingSchoolName("National Institute of Dramatic Art");
        return movieActor;
    }
}

```

Die DynamoDB-Tabellen enthalten die folgenden Elemente, bevor das Codebeispiel ausgeführt wird.

```

1 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2 | MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2013, actingSchoolName='National Institute of Dramatic Art'}

```

Die folgenden Elemente befinden sich in den Tabellen, nachdem die Ausführung des Codes abgeschlossen ist.

```

ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2023-03-15 11:29:07 [main] INFO org.example.tests.TransactDemoTest:168 -
MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2013, actingSchoolName='National Institute of Dramatic Art'}

```

Elemente bleiben in den Tabellen unverändert, da die Transaktion fehlgeschlagen ist. Der `actingYear` Wert für den Film `Blue Jasmine` ist `2013` wie in Zeile 2 in der Liste der Elemente dargestellt, bevor das Codebeispiel ausgeführt wird.

Die folgenden Zeilen werden in der -Konsole protokolliert.

```
CancellationReason(Code=None)
CancellationReason(Code=None)
CancellationReason(Item={actor=AttributeValue(S=Cate Blanchett),
  movie=AttributeValue(S=Blue Jasmine), actingaward=AttributeValue(S=Best Actress),
  actingyear=AttributeValue(N=2013), actingschoolname=AttributeValue(S=National
  Institute of Dramatic Art)},
  Code=ConditionalCheckFailed, Message=The conditional request failed)
```

## Sekundäre Indizes verwenden

Sekundäre Indizes verbessern den Datenzugriff, indem alternative Schlüssel definiert werden, die Sie bei Abfrage- und Scanvorgängen verwenden. Globale sekundäre Indizes (GSI) haben einen Partitionsschlüssel und einen Sortierschlüssel, die sich von denen in der Basistabelle unterscheiden können. Im Gegensatz dazu verwenden lokale sekundäre Indizes (LSI) den Partitionsschlüssel des Primärindex.

Datenklasse mit sekundären Index-Anmerkungen kommentieren

Attribute, die an sekundären Indizes beteiligt sind, erfordern entweder die -@DynamoDbSecondaryPartitionKey oder die -@DynamoDbSecondarySortKeyAnmerkung.

Die folgende Klasse zeigt Anmerkungen für zwei Indizes. Der GSI mit dem Namen SubjectLastPostedDateIndex verwendet das -SubjectAttribut für den Partitionsschlüssel und den LastPostedDateTime für den Sortierschlüssel. Der LSI mit dem Namen ForumLastPostedDateIndex verwendet ForumName als Partitionsschlüssel und LastPostedDateTime als Sortierschlüssel.

Beachten Sie, dass das Subject Attribut eine duale Rolle erfüllt. Es ist der Sortierschlüssel des Primärschlüssels und der Partitionsschlüssel des GSI mit dem Namen SubjectLastPostedDateIndex.

## MessageThread-Klasse

Die MessageThread Klasse eignet sich für die Verwendung als Datenklasse für die [Beispieltabelle Thread](#) im Amazon-DynamoDB-Entwicklerhandbuch.

## Importe

```
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbBean;
```

```
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondarySortKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.util.List;
```

```
@DynamoDbBean
public class MessageThread {
    private String ForumName;
    private String Subject;
    private String Message;
    private String LastPostedBy;
    private String LastPostedDateTime;
    private Integer Views;
    private Integer Replies;
    private Integer Answered;
    private List<String> Tags;

    @DynamoDbPartitionKey
    public String getForumName() {
        return ForumName;
    }

    public void setForumName(String forumName) {
        ForumName = forumName;
    }

    // Sort key for primary index and partition key for GSI
    "SubjectLastPostedDateIndex".
    @DynamoDbSortKey
    @DynamoDbSecondaryPartitionKey(indexNames = "SubjectLastPostedDateIndex")
    public String getSubject() {
        return Subject;
    }

    public void setSubject(String subject) {
        Subject = subject;
    }
}
```



```
// Sort key for GSI "SubjectLastPostedDateIndex" and sort key for LSI
"ForumLastPostedDateIndex".
    @DynamoDbSecondarySortKey(indexNames = {"SubjectLastPostedDateIndex",
"ForumLastPostedDateIndex"})
    public String getLastPostedDateTime() {
        return LastPostedDateTime;
    }

    public void setLastPostedDateTime(String lastPostedDateTime) {
        LastPostedDateTime = lastPostedDateTime;
    }

    public String getMessage() {
        return Message;
    }

    public void setMessage(String message) {
        Message = message;
    }

    public String getLastPostedBy() {
        return LastPostedBy;
    }

    public void setLastPostedBy(String lastPostedBy) {
        LastPostedBy = lastPostedBy;
    }

    public Integer getViews() {
        return Views;
    }

    public void setViews(Integer views) {
        Views = views;
    }

    @DynamoDbSecondaryPartitionKey(indexNames = "ForumRepliesIndex")
    public Integer getReplies() {
        return Replies;
    }

    public void setReplies(Integer replies) {
        Replies = replies;
    }
}
```

```
public Integer getAnswered() {
    return Answered;
}

public void setAnswered(Integer answered) {
    Answered = answered;
}

public List<String> getTags() {
    return Tags;
}

public void setTags(List<String> tags) {
    Tags = tags;
}

public MessageThread() {
    this.Answered = 0;
    this.LastPostedBy = "";
    this.ForumName = "";
    this.Message = "";
    this.LastPostedDateTime = "";
    this.Replies = 0;
    this.Views = 0;
    this.Subject = "";
}

@Override
public String toString() {
    return "MessageThread{" +
        "ForumName='" + ForumName + '\'' +
        ", Subject='" + Subject + '\'' +
        ", Message='" + Message + '\'' +
        ", LastPostedBy='" + LastPostedBy + '\'' +
        ", LastPostedDateTime='" + LastPostedDateTime + '\'' +
        ", Views=" + Views +
        ", Replies=" + Replies +
        ", Answered=" + Answered +
        ", Tags=" + Tags +
        '}';
}
}
```

## Erstellen des Index

Ab Version 2.20.86 des SDK for Java generiert die `createTable()` Methode automatisch sekundäre Indizes aus Datenklassenmerkungen. Standardmäßig werden alle Attribute aus der Basistabelle in einen Index kopiert und die bereitgestellten Durchsatzwerte betragen 20 Lesekapazitätseinheiten und 20 Schreibkapazitätseinheiten.

Wenn Sie jedoch eine SDK-Version vor dem 2.20.86 verwenden, müssen Sie den Index zusammen mit der Tabelle erstellen, wie im folgenden Beispiel gezeigt. In diesem Beispiel werden die beiden Indizes für die Thread Tabelle erstellt. Der [Builder](#)-Parameter verfügt über Methoden zum Konfigurieren beider Indizes, wie nach den Kommentarzeilen 1 und 2 gezeigt. Sie verwenden die `-indexName()` Methode des Index Builders, um die in den Datenklassenmerkungen angegebenen Indexnamen dem beabsichtigten Indextyp zuzuordnen.

Dieser Code konfiguriert alle Tabellenattribute so, dass sie nach den Kommentarzeilen 3 und 4 in beiden Indizes landen. Weitere Informationen zu [Attributprojektionen finden Sie im Amazon-DynamoDB-Entwicklerhandbuch](#).

```
public static void createMessageThreadTable(DynamoDbTable<MessageThread>
messageThreadDynamoDbTable, DynamoDbClient dynamoDbClient) {
    messageThreadDynamoDbTable.createTable(b -> b
        // 1. Generate the GSI.
        .globalSecondaryIndices(gsi ->
gsi.indexName("SubjectLastPostedDateIndex")
        // 3. Populate the GSI with all attributes.
        .projection(p -> p
            .projectionType(ProjectionType.ALL))
        )
        // 2. Generate the LSI.
        .localSecondaryIndices(lsi -> lsi.indexName("ForumLastPostedDateIndex")
        // 4. Populate the LSI with all attributes.
        .projection(p -> p
            .projectionType(ProjectionType.ALL))
        )
    );
}
```

## Abfragen mithilfe eines Index

Im folgenden Beispiel wird der lokale sekundäre Index abgefragt `ForumLastPostedDateIndex`.

Im Anschluss an die Kommentarzeile 2 erstellen Sie ein [QueryConditional](#) Objekt, das beim Aufrufen der Methode [DynamoDbIndex.query\(\)](#) erforderlich ist.

Sie erhalten einen Verweis auf den Index, den Sie nach der Kommentarzeile 3 abfragen möchten, indem Sie den Namen des Index übergeben. Nach Kommentarzeile 4 rufen Sie die `-query()` Methode für den Index auf, der das `-QueryConditional` Objekt übergibt.

Sie konfigurieren die Abfrage auch so, dass drei Attributwerte zurückgegeben werden, wie nach der Kommentarzeile 5 gezeigt. Wenn nicht aufgerufen `attributesToProject()` wird, gibt die Abfrage alle Attributwerte zurück. Beachten Sie, dass die angegebenen Attributnamen mit Kleinbuchstaben beginnen. Diese Attributnamen stimmen mit denen überein, die in der Tabelle verwendet werden, nicht unbedingt mit den Attributnamen der Datenklasse.

Im Anschluss an Kommentarzeile 6 durchlaufen Sie die Ergebnisse und protokollieren jedes von der Abfrage zurückgegebene Element und speichern es auch in der Liste, um zum Aufrufer zurückzukehren.

```
public static List<MessageThread> queryUsingSecondaryIndices(DynamoDbEnhancedClient
    enhancedClient,
                                                                String lastPostedDate,
                                                                DynamoDbTable<MessageThread> threadTable) {
    // 1. Log the parameter value.
    logger.info("lastPostedDate value: {}", lastPostedDate);

    // 2. Create a QueryConditional whose sort key value must be greater than or
    equal to the parameter value.
    QueryConditional queryConditional =
    QueryConditional.sortGreaterThanOrEqualTo(qc ->
        qc.partitionValue("Forum02").sortValue(lastPostedDate));

    // 3. Specify the index name to query the DynamoDbIndex instance.
    final DynamoDbIndex<MessageThread> forumLastPostedDateIndex =
    threadTable.index("ForumLastPostedDateIndex");

    // 4. Perform the query by using the QueryConditional object.
    final SdkIterable<Page<MessageThread>> pagedResult =
    forumLastPostedDateIndex.query(q -> q
        .queryConditional(queryConditional)
        // 5. Request three attribute in the results.
        .attributesToProject("forumName", "subject", "lastPostedDateTime"));

    List<MessageThread> collectedItems = new ArrayList<>();
    // 6. Iterate through the pages response and sort the items.
    pagedResult.stream().forEach(page -> page.items().stream())
```

```
.sorted(Comparator.comparing(MessageThread::getLastPostedDateTime))
    .forEach(mt -> {
        // 7. Log the returned items and add the collection to
return to the caller.
        logger.info(mt.toString());
        collectedItems.add(mt);
    });
return collectedItems;
}
```

Die folgenden Elemente sind in der Datenbank vorhanden, bevor die Abfrage ausgeführt wird.

```
MessageThread{ForumName='Forum01', Subject='Subject01', Message='Message01',
  LastPostedBy='', LastPostedDateTime='2023.03.28', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject02', Message='Message02',
  LastPostedBy='', LastPostedDateTime='2023.03.29', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject04', Message='Message04',
  LastPostedBy='', LastPostedDateTime='2023.03.31', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject08', Message='Message08',
  LastPostedBy='', LastPostedDateTime='2023.04.04', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject10', Message='Message10',
  LastPostedBy='', LastPostedDateTime='2023.04.06', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum03', Subject='Subject03', Message='Message03',
  LastPostedBy='', LastPostedDateTime='2023.03.30', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum03', Subject='Subject06', Message='Message06',
  LastPostedBy='', LastPostedDateTime='2023.04.02', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum03', Subject='Subject09', Message='Message09',
  LastPostedBy='', LastPostedDateTime='2023.04.05', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum05', Subject='Subject05', Message='Message05',
  LastPostedBy='', LastPostedDateTime='2023.04.01', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum07', Subject='Subject07', Message='Message07',
  LastPostedBy='', LastPostedDateTime='2023.04.03', Views=0, Replies=0, Answered=0,
  Tags=null}
```

Die Protokollierungsanweisungen in den Zeilen 1 und 6 führen zu der folgenden Konsolenausgabe.

```
lastPostedDate value: 2023.03.31
MessageThread{ForumName='Forum02', Subject='Subject04', Message='', LastPostedBy='',
  LastPostedDateTime='2023.03.31', Views=0, Replies=0, Answered=0, Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject08', Message='', LastPostedBy='',
  LastPostedDateTime='2023.04.04', Views=0, Replies=0, Answered=0, Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject10', Message='', LastPostedBy='',
  LastPostedDateTime='2023.04.06', Views=0, Replies=0, Answered=0, Tags=null}
```

Die Abfrage hat Elemente mit dem `forumName` Wert `Forum02` und einem `lastPostedDateTime` Wert größer oder gleich `2023.03.31` zurückgegeben. Die Ergebnisse zeigen `message` Werte mit einer leeren Zeichenfolge, obwohl die `message` Attribute Werte im Index haben. Dies liegt daran, dass das Nachrichtenattribut nach der Kommentarzeile 5 nicht projiziert wurde.

## Verwenden Sie erweiterte Mapping-Funktionen

Erfahren Sie mehr über erweiterte Tabellenschemafunktionen in der DynamoDB Enhanced Client API.

Machen Sie sich mit Tabellenschematypen vertraut

[TableSchema](#) ist die Schnittstelle zur Mapping-Funktionalität der DynamoDB Enhanced Client API. Sie kann ein Datenobjekt einer Map von und zu einer Map von zuordnen. [AttributeValues](#) Ein `TableSchema` Objekt muss die Struktur der Tabelle kennen, die es abbildet. Diese Strukturinformationen werden in einem [TableMetadata](#) Objekt gespeichert.

Die erweiterte Client-API hat mehrere Implementierungen von `TableSchema`, die im Folgenden aufgeführt sind.

Aus kommentierten Klassen generiertes Tabellenschema

Das Erstellen eines `TableSchema` aus annotierten Klassen ist ein relativ teurer Vorgang. Wir empfehlen daher, dies einmal beim Start der Anwendung zu tun.

### [BeanTableSchema](#)

Diese Implementierung basiert auf Attributen und Anmerkungen einer Bean-Klasse. Ein Beispiel für diesen Ansatz wird im [Abschnitt Erste Schritte](#) demonstriert.

**Note**

Wenn `BeanTableSchema` sich nicht wie erwartet verhält, aktivieren Sie die Debug-Protokollierung für `software.amazon.awssdk.enhanced.dynamodb.beans`

## [ImmutableTableSchema](#)

Diese Implementierung basiert auf einer unveränderlichen Datenklasse. Dieser Ansatz wird im [??? Abschnitt](#) beschrieben.

Mit einem Builder generiertes Tabellenschema

Die folgenden `TableSchema`s werden mithilfe eines Builders aus Code erstellt. Dieser Ansatz ist kostengünstiger als der Ansatz, bei dem annotierte Datenklassen verwendet werden. Der Builder-Ansatz vermeidet die Verwendung von Anmerkungen und erfordert keine JavaBean Benennungsstandards.

## [StaticTableSchema](#)

Diese Implementierung wurde für veränderbare Datenklassen entwickelt. Im Abschnitt „Erste Schritte“ dieses Handbuchs wurde gezeigt, wie Sie [StaticTableSchema mithilfe eines Builders eine generieren](#).

## [StaticImmutableTableSchema](#)

Ähnlich wie beim Erstellen eines generieren Sie eine Implementierung dieses `TypesStaticTableSchema`, indem Sie einen [Builder](#) für die `TableSchema` Verwendung mit unveränderlichen Datenklassen verwenden.

Tabellenschema für Daten ohne festes Schema

## [DocumentTableSchema](#)

Im Gegensatz zu anderen Implementierungen von `TableSchema` definieren Sie keine Attribute für eine `DocumentTableSchema` Instanz. Normalerweise geben Sie nur Primärschlüssel und Anbieter von Attributkonvertern an. Eine `EnhancedDocument` Instanz stellt die Attribute bereit, die Sie aus einzelnen Elementen oder aus einer JSON-Zeichenfolge erstellen.

## Fügen Sie Attribute explizit ein oder schließen Sie sie aus

Die DynamoDB Enhanced Client API bietet Anmerkungen, um Datenklassenattribute daran zu hindern, zu Attributen in einer Tabelle zu werden. Mit der API können Sie auch einen Attributnamen verwenden, der sich vom Attributnamen der Datenklasse unterscheidet.

### Attribute ausschließen

Um Attribute zu ignorieren, die keiner DynamoDB-Tabelle zugeordnet werden sollten, markieren Sie das Attribut mit der Anmerkung. `@DynamoDbIgnore`

```
private String internalKey;

@DynamoDbIgnore
public String getInternalKey() { return this.internalKey; }
public void setInternalKey(String internalKey) { return this.internalKey =
    internalKey;}
```

### Attribute einbeziehen

Um den Namen eines in der DynamoDB-Tabelle verwendeten Attributs zu ändern, markieren Sie es mit der `@DynamoDbAttribute` Anmerkung und geben Sie einen anderen Namen ein.

```
private String internalKey;

@DynamoDbAttribute("renamedInternalKey")
public String getInternalKey() { return this.internalKey; }
public void setInternalKey(String internalKey) { return this.internalKey =
    internalKey;}
```

### Steuern Sie die Attributkonvertierung

Standardmäßig stellt ein Tabellenschema über eine Standardimplementierung der [AttributeConverterProvider](#) Schnittstelle Konverter für alle primitiven Typen und viele gängige Java-Typen bereit. Sie können das allgemeine Standardverhalten mit einer benutzerdefinierten `AttributeConverterProvider` Implementierung ändern. Sie können den Konverter auch für ein einzelnes Attribut ändern.

Eine Liste der verfügbaren Konverter finden Sie in der [AttributeConverter](#) Java-Dokumentation zur Schnittstelle.



## Stellen Sie Anbieter für benutzerdefinierte Attributkonverter bereit

Sie können über die `@DynamoDbBean (converterProviders = {...})` Anmerkung ein einzelnes `AttributeConverterProvider` oder eine Kette von bestellten `AttributeConverterProvider` S angeben. Jeder benutzerdefinierte `AttributeConverterProvider` Benutzer muss die `AttributeConverterProvider` Schnittstelle erweitern.

Beachten Sie, dass Sie, wenn Sie Ihre eigene Kette von Anbietern für Attributkonverter angeben, den Standardkonverter-Anbieter außer Kraft setzen `DefaultAttributeConverterProvider`. Wenn Sie die Funktionalität von verwenden möchten `DefaultAttributeConverterProvider`, müssen Sie sie in die Kette aufnehmen.

Es ist auch möglich, die Bean mit einem leeren Array `{}` zu annotieren. Dadurch wird die Verwendung aller Anbieter für Attributkonverter deaktiviert, einschließlich der Standardanbieter. In diesem Fall müssen alle Attribute, die zugeordnet werden sollen, über einen eigenen Attributkonverter verfügen.

Das folgende Snippet zeigt einen einzelnen Konverter-Anbieter.

```
@DynamoDbBean(converterProviders = ConverterProvider1.class)
public class Customer {

}
```

Der folgende Ausschnitt zeigt die Verwendung einer Kette von Konverteranbietern. Da der SDK-Standard zuletzt bereitgestellt wird, hat er die niedrigste Priorität.

```
@DynamoDbBean(converterProviders = {
    ConverterProvider1.class,
    ConverterProvider2.class,
    DefaultAttributeConverterProvider.class})
public class Customer {

}
```

Die Schema-BUILDER für statische Tabellen verfügen über eine `attributeConverterProviders()` Methode, die auf die gleiche Weise funktioniert. Dies wird im folgenden Snippet gezeigt.

```
private static final StaticTableSchema<Customer> CUSTOMER_TABLE_SCHEMA =
    StaticTableSchema.builder(Customer.class)
```

```

.newItemSupplier(Customer::new)
.addAttribute(String.class, a -> a.name("name")
                a.getter(Customer::getName)
                a.setter(Customer::setName))
.attributeConverterProviders(converterProvider1, converterProvider2)
.build();

```

## Überschreiben Sie die Zuordnung eines einzelnen Attributs

Um die Art und Weise, wie ein einzelnes Attribut zugeordnet wird, zu überschreiben, geben Sie an `AttributeConverter` für das Attribut ein. Dieser Zusatz hat Vorrang vor allen Konvertern, die von `AttributeConverterProviders` im Tabellenschema bereitgestellt werden. Dadurch wird ein benutzerdefinierter Konverter nur für dieses Attribut hinzugefügt. Andere Attribute, auch solche desselben Typs, verwenden diesen Konverter nur, wenn er explizit für diese anderen Attribute angegeben ist.

Die `@DynamoDbConvertedBy` Anmerkung wird verwendet, um die benutzerdefinierte `AttributeConverter` Klasse anzugeben, wie im folgenden Codeausschnitt gezeigt.

```

@DynamoDbBean
public class Customer {
    private String name;

    @DynamoDbConvertedBy(CustomAttributeConverter.class)
    public String getName() { return this.name; }
    public void setName(String name) { this.name = name;}
}

```

Die Builder für statische Schemas verfügen über eine entsprechende Methode zur Erstellung von Attributen. `attributeConverter()` Diese Methode benötigt eine Instanz von `a`, `AttributeConverter` wie im Folgenden gezeigt.

```

private static final StaticTableSchema<Customer> CUSTOMER_TABLE_SCHEMA =
    StaticTableSchema.builder(Customer.class)
        .newItemSupplier(Customer::new)
        .addAttribute(String.class, a -> a.name("name")
                    a.getter(Customer::getName)
                    a.setter(Customer::setName)
                    a.attributeConverter(customAttributeConverter))
        .build();

```

## Beispiel

Dieses Beispiel zeigt eine `AttributeConverterProvider` Implementierung, die einen Attributkonverter für [java.net.HttpCookie](#) Objekte bereitstellt.

Die folgende `SimpleUser` Klasse enthält ein Attribut mit dem Namen `lastUsedCookie`, das eine Instanz von `HttpCookie` ist.

Der Parameter für die `@DynamoDbBean` Anmerkungen listet die beiden `AttributeConverterProvider` Klassen auf, die Konverter bereitstellen.

### Class with annotations

```
@DynamoDbBean(converterProviders = {CookieConverterProvider.class,
DefaultAttributeConverterProvider.class})
public static final class SimpleUser {
    private String name;
    private HttpCookie lastUsedCookie;

    @DynamoDbPartitionKey
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public HttpCookie getLastUsedCookie() {
        return lastUsedCookie;
    }

    public void setLastUsedCookie(HttpCookie lastUsedCookie) {
        this.lastUsedCookie = lastUsedCookie;
    }
}
```

### Static table schema

```
private static final TableSchema<SimpleUser> SIMPLE_USER_TABLE_SCHEMA =
    TableSchema.builder(SimpleUser.class)
        .newItemSupplier(SimpleUser::new)
        .attributeConverterProviders(CookieConverterProvider.create(),
AttributeConverterProvider.defaultProvider())
```

```

        .addAttribute(String.class, a -> a.name("name")
            .setter(SimpleUser::setName)
            .getter(SimpleUser::getName)
            .tags(StaticAttributeTags.primaryPartitionKey()))
        .addAttribute(HttpCookie.class, a -> a.name("lastUsedCookie")
            .setter(SimpleUser::setLastUsedCookie)
            .getter(SimpleUser::getLastUsedCookie))
        .build();

```

Das `CookieConverterProvider` folgende Beispiel bietet eine Instanz `HttpCookieConverter` von.

```

public static final class CookieConverterProvider implements
AttributeConverterProvider {
    private final Map<EnhancedType<?>, AttributeConverter<?>> converterCache =
ImmutableMap.of(
        // 1. Add HttpCookieConverter to the internal cache.
        EnhancedType.of(HttpCookie.class), new HttpCookieConverter());

    public static CookieConverterProvider create() {
        return new CookieConverterProvider();
    }

    // The SDK calls this method to find out if the provider contains a
AttributeConverter instance
    // for the EnhancedType<T> argument.
    @SuppressWarnings("unchecked")
    @Override
    public <T> AttributeConverter<T> converterFor(EnhancedType<T> enhancedType) {
        return (AttributeConverter<T>) converterCache.get(enhancedType);
    }
}

```

## Konvertierungscode

In der `transformFrom()` Methode der folgenden `HttpCookieConverter` Klasse empfängt der Code eine `HttpCookie` Instanz und wandelt sie in eine `DynamoDB-Map` um, die als Attribut gespeichert wird.

Die `transformTo()` Methode empfängt einen `DynamoDB-Zuordnungsparameter` und ruft dann den `HttpCookie` Konstruktor auf, der einen Namen und einen Wert benötigt.

```
public static final class HttpCookieConverter implements
AttributeConverter<HttpCookie> {

    @Override
    public AttributeValue transformFrom(HttpCookie httpCookie) {

        return AttributeValue.fromM(
            Map.of ("cookieName", AttributeValue.fromS(httpCookie.getName()),
                "cookieValue", AttributeValue.fromS(httpCookie.getValue()))
        );
    }

    @Override
    public HttpCookie transformTo(AttributeValue attributeValue) {
        Map<String, AttributeValue> map = attributeValue.m();
        return new HttpCookie(
            map.get("cookieName").s(),
            map.get("cookieValue").s());
    }

    @Override
    public EnhancedType<HttpCookie> type() {
        return EnhancedType.of(HttpCookie.class);
    }

    @Override
    public AttributeValueType attributeValueType() {
        return AttributeValueType.M;
    }
}
```

## Ändern Sie das Aktualisierungsverhalten von Attributen

Sie können das Aktualisierungsverhalten einzelner Attribute anpassen, wenn Sie einen Aktualisierungsvorgang durchführen. [Einige Beispiele für Aktualisierungsvorgänge in der DynamoDB Enhanced Client API sind `updateItem\(\)` und `transactWriteItems\(\)`](#).

Stellen Sie sich zum Beispiel vor, Sie möchten einen am erstellten Zeitstempel in Ihrem Datensatz speichern. Sie möchten jedoch, dass sein Wert nur geschrieben wird, wenn für das Attribut noch kein Wert in der Datenbank vorhanden ist. In diesem Fall verwenden Sie das [WRITE\\_IF\\_NOT\\_EXISTS](#) Aktualisierungsverhalten.

Das folgende Beispiel zeigt die Anmerkung, die dem `createdOn` Attribut das Verhalten hinzufügt.

```
@DynamoDbBean
public class Customer extends GenericRecord {
    private String id;
    private Instant createdOn;

    @DynamoDbPartitionKey
    public String getId() { return this.id; }
    public void setId(String id) { this.name = id; }

    @DynamoDbUpdateBehavior(UpdateBehavior.WRITE_IF_NOT_EXISTS)
    public Instant getCreatedOn() { return this.createdOn; }
    public void setCreatedOn(Instant createdOn) { this.createdOn = createdOn; }
}
```

Sie können dasselbe Aktualisierungsverhalten deklarieren, wenn Sie ein statisches Tabellenschema erstellen, wie im folgenden Beispiel nach Kommentarzeile 1 gezeigt.

```
static final TableSchema<Customer> CUSTOMER_TABLE_SCHEMA =
    TableSchema.builder(Customer.class)
        .newItemSupplier(Customer::new)
        .addAttribute(String.class, a -> a.name("id")
            .getter(Customer::getId)
            .setter(Customer::setId)

        .tags(StaticAttributeTags.primaryPartitionKey()))
        .addAttribute(Instant.class, a -> a.name("createdOn")
            .getter(Customer::getCreatedOn)
            .setter(Customer::setCreatedOn)
            // 1. Add an UpdateBehavior.

        .tags(StaticAttributeTags.updateBehavior(UpdateBehavior.WRITE_IF_NOT_EXISTS)))
        .build();
```

## Reduzieren Sie Attribute aus anderen Klassen

Wenn die Attribute für Ihre Tabelle entweder durch Vererbung oder Zusammensetzung auf mehrere verschiedene Java-Klassen verteilt sind, bietet die DynamoDB Enhanced Client-API Unterstützung, um die Attribute zu einer Klasse zusammenzufassen.

## Verwenden Sie Vererbung

Wenn Ihre Klassen Vererbung verwenden, verwenden Sie die folgenden Methoden, um die Hierarchie zu vereinfachen.

## Verwenden Sie Beans mit Anmerkungen

Für den Annotationsansatz müssen beide Klassen die `@DynamoDbBean` Annotation enthalten und eine Klasse muss eine oder mehrere Primärschlüssel-Annotationen enthalten.

Im Folgenden werden Beispiele für Datenklassen gezeigt, die eine Vererbungsbeziehung haben.

### Standard data class

```
@DynamoDbBean
public class Customer extends GenericRecord {
    private String name;

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}

@dynamoDbBean
public abstract class GenericRecord {
    private String id;
    private String createdAt;

    @DynamoDbPartitionKey
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedAt() { return createdAt; }
    public void setCreatedAt(String createdAt) { this.createdAt =
    createdAt; }
}
```

## Lombok

Die [onMethodOption](#) von Lombok kopiert attributbasierte DynamoDB-Anmerkungen, wie z. B., in den generierten Code. `@DynamoDbPartitionKey`

```
@DynamoDbBean
```

```

@Data
@ToString(callSuper = true)
public class Customer extends GenericRecord {
    private String name;
}

@Data
@DynamoDbBean
public abstract class GenericRecord {
    @Getter(onMethod_=@DynamoDbPartitionKey)
    private String id;
    private String createdAt;
}

```

## Verwenden Sie statische Schemas

Verwenden Sie für den statischen Schemaansatz die `extend()` Methode des Builders, um die Attribute der übergeordneten Klasse auf die untergeordnete Klasse zu reduzieren. Dies wird im folgenden Beispiel nach Kommentarzeile 1 gezeigt.

```

        StaticTableSchema<org.example.tests.model.inheritance.stat.GenericRecord>
GENERIC_RECORD_SCHEMA =

StaticTableSchema.builder(org.example.tests.model.inheritance.stat.GenericRecord.class)
    // The partition key will be inherited by the top level mapper.
    .addAttribute(String.class, a -> a.name("id"))

    .getter(org.example.tests.model.inheritance.stat.GenericRecord::getId)

    .setter(org.example.tests.model.inheritance.stat.GenericRecord::setId)
        .tags(primaryPartitionKey())
        .addAttribute(String.class, a -> a.name("created_date"))

    .getter(org.example.tests.model.inheritance.stat.GenericRecord::getCreatedAt)

    .setter(org.example.tests.model.inheritance.stat.GenericRecord::setCreatedAt))
        .build();

        StaticTableSchema<org.example.tests.model.inheritance.stat.Customer>
CUSTOMER_SCHEMA =

StaticTableSchema.builder(org.example.tests.model.inheritance.stat.Customer.class)

```



```

.newItemSupplier(org.example.tests.model.inheritance.stat.Customer::new)
    .addAttribute(String.class, a -> a.name("name"))

.getter(org.example.tests.model.inheritance.stat.Customer::getName)

.setter(org.example.tests.model.inheritance.stat.Customer::setName))
    // 1. Use the extend() method to collapse the parent attributes
onto the child class.
    .extend(GENERIC_RECORD_SCHEMA) // All the attributes of the
GenericRecord schema are added to Customer.
    .build();

```

Das vorherige Beispiel für ein statisches Schema verwendet die folgenden Datenklassen. Da die Zuordnung beim Erstellen des statischen Tabellenschemas definiert wird, benötigen die Datenklassen keine Anmerkungen.

## Datenklassen

### Standard data class

```

public class Customer extends GenericRecord {
    private String name;

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}

public abstract class GenericRecord {
    private String id;
    private String createdAt;

    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedAt() { return createdAt; }
    public void setCreatedAt(String createdAt) { this.createdAt =
createdAt; }
}

```

## Lombok

```
@Data
```

```
@ToString(callSuper = true)
public class Customer extends GenericRecord{
    private String name;
}

@Data
public abstract class GenericRecord {
    private String id;
    private String createdAt;
}
```

## Verwenden Sie die Zusammensetzung

Wenn in Ihren Klassen Komposition verwendet wird, verwenden Sie die folgenden Methoden, um die Hierarchie zu vereinfachen.

## Verwenden Sie Beans mit Anmerkungen

Die `@DynamoDbFlatten` Anmerkung reduziert die enthaltene Klasse.

In den folgenden Beispielen für Datenklassen wird die `@DynamoDbFlatten` Anmerkung verwendet, um der Klasse effektiv alle Attribute der enthaltenen `GenericRecord` Klasse hinzuzufügen.

### Customer

#### Standard data class

```
@DynamoDbBean
public class Customer {
    private String name;
    private GenericRecord record;

    public String getName() { return this.name; }
    public void setName(String name) { this.name = name; }

    @DynamoDbFlatten
    public GenericRecord getRecord() { return this.record; }
    public void setRecord(GenericRecord record) { this.record = record; }

    @DynamoDbBean
    public class GenericRecord {
        private String id;
        private String createdAt;
    }
}
```

```
@DynamoDbPartitionKey
public String getId() { return this.id; }
public void setId(String id) { this.id = id; }

public String getCreateDate() { return this.createDate; }
public void setCreateDate(String createDate) { this.createDate =
createDate; }
}
```

## Lombok

```
@Data
@DynamoDbBean
public class Customer {
    private String name;
    @Getter(onMethod_=@DynamoDbFlatten)
    private GenericRecord record;
}

@Data
@DynamoDbBean
public class GenericRecord {
    @Getter(onMethod_=@DynamoDbPartitionKey)
    private String id;
    private String createDate;
}
```

Sie können die Annotation `Flatten` verwenden, um so viele verschiedene geeignete Klassen wie nötig zu reduzieren. Die folgenden Einschränkungen gelten:

- Alle Attributnamen müssen eindeutig sein, nachdem sie reduziert wurden.
- Es darf nie mehr als einen Partitionsschlüssel, Sortierschlüssel oder Tabellennamen geben.

## Verwenden Sie statische Schemas

Wenn Sie ein statisches Tabellenschema erstellen, verwenden Sie die `flatten()` Methode des Builders. Sie stellen auch die Getter- und Setter-Methoden bereit, die die enthaltene Klasse identifizieren.

```

StaticTableSchema<GenericRecord> GENERIC_RECORD_SCHEMA =
    StaticTableSchema.builder(GenericRecord.class)
        .newItemSupplier(GenericRecord::new)
        .addAttribute(String.class, a -> a.name("id")
            .getter(GenericRecord::getId)
            .setter(GenericRecord::setId)
            .tags(primaryPartitionKey()))
        .addAttribute(String.class, a -> a.name("created_date")
            .getter(GenericRecord::getCreatedDate)
            .setter(GenericRecord::setCreatedDate))
        .build();

StaticTableSchema<Customer> CUSTOMER_SCHEMA =
    StaticTableSchema.builder(Customer.class)
        .newItemSupplier(Customer::new)
        .addAttribute(String.class, a -> a.name("name")
            .getter(Customer::getName)
            .setter(Customer::setName))
        // Because we are flattening a component object, we supply a
getter and setter so the
        // mapper knows how to access it.
        .flatten(GENERIC_RECORD_SCHEMA, Customer::getRecord,
Customer::setRecord)
        .build();

```

Das vorherige Beispiel für ein statisches Schema verwendet die folgenden Datenklassen.

## Datenklassen

### Standard data class

```

public class Customer {
    private String name;
    private GenericRecord record;

    public String getName() { return this.name; }
    public void setName(String name) { this.name = name; }

    public GenericRecord getRecord() { return this.record; }
    public void setRecord(GenericRecord record) { this.record = record; }

public class GenericRecord {
    private String id;

```

```

private String createdAt;

public String getId() { return this.id; }
public void setId(String id) { this.id = id; }

public String getCreatedAt() { return this.createdAt; }
public void setCreatedAt(String createdAt) { this.createdAt =
createdAt; }
}

```

## Lombok

```

@Data
public class Customer {
    private String name;
    private GenericRecord record;
}

@Data
public class GenericRecord {
    private String id;
    private String createdAt;
}

```

Sie können das Builder-Muster verwenden, um so viele verschiedene in Frage kommende Klassen zu reduzieren, wie Sie möchten.

### Implikationen für anderen Code

Wenn Sie das `@DynamoDbFlatten` Attribut (oder die `flatten()` Builder-Methode) verwenden, enthält das Element in DynamoDB ein Attribut für jedes Attribut des erstellten Objekts. Es enthält auch die Attribute des zusammengesetzten Objekts.

Wenn Sie dagegen eine Datenklasse mit einer zusammengesetzten Klasse annotieren und diese nicht verwenden `@DynamoDbFlatten`, wird das Element zusammen mit dem zusammengesetzten Objekt als einzelnes Attribut gespeichert.

Vergleichen Sie beispielsweise die im [Beispiel „Reduzieren mit Komposition“](#) gezeigte [Customer Klasse mit](#) und ohne Reduzierung des Attributs `record`. Sie können den Unterschied mit JSON visualisieren, wie in der folgenden Tabelle dargestellt.

Mit Abflachen	Ohne Abflachen
3 Attribute	2 Attribute
<pre>{   "id": "1",   "createdDate": "today",   "name": "my name" }</pre>	<pre>{   "id": "1",   "record": {     "createdDate": "today",     "name": "my name"   } }</pre>

Der Unterschied wird wichtig, wenn Sie anderen Code haben, der auf die DynamoDB-Tabelle zugreift und erwartet, bestimmte Attribute zu finden.

Arbeiten Sie mit verschachtelten Attributen

Ein verschachteltes Attribut in DynamoDB ist in ein anderes Attribut eingebettet. Beispiele hierfür sind Listenelemente und Karteneinträge.

In Java entspricht ein verschachteltes DynamoDB-Attribut einem Mitglied einer Klasse, das ein oder ist. List Map Es entspricht auch einer Instanz eines komplexen Typs wie Address oder PhoneNumber, wie er in der folgenden Klasse verwendet wird. Person

### Person-Klasse

```
@DynamoDbBean
public class Person {
    Integer id;
    String firstName;
    String lastName;
    Integer age;
    Map<String, Address> addresses;
    List<PhoneNumber> phoneNumbers;

    List<String> hobbies;

    @DynamoDbPartitionKey
    public Integer getId() {
        return id;
    }
}
```

```
}

public void setId(Integer id) {
    this.id = id;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public Integer getAge() {
    return age;
}

public void setAge(Integer age) {
    this.age = age;
}

public Map<String, Address> getAddresses() {
    return addresses;
}

public void setAddresses(Map<String, Address> addresses) {
    this.addresses = addresses;
}

public List<PhoneNumber> getPhoneNumbers() {
    return phoneNumbers;
}

public void setPhoneNumbers(List<PhoneNumber> phoneNumbers) {
    this.phoneNumbers = phoneNumbers;
}
```

```
    }

    public List<String> getHobbies() {
        return hobbies;
    }

    public void setHobbies(List<String> hobbies) {
        this.hobbies = hobbies;
    }

    @Override
    public String toString() {
        return "Person{" +
            "id=" + id +
            ", firstName='" + firstName + '\'' +
            ", lastName='" + lastName + '\'' +
            ", age=" + age +
            ", addresses=" + addresses +
            ", phoneNumbers=" + phoneNumbers +
            ", hobbies=" + hobbies +
            '}';
    }
}
```

## Address-Klasse

```
@DynamoDbBean
public class Address {
    private String street;
    private String city;
    private String state;
    private String zipCode;

    public Address() {
    }

    public String getStreet() {
        return this.street;
    }

    public String getCity() {
        return this.city;
    }
}
```



```
public String getState() {
    return this.state;
}

public String getZipCode() {
    return this.zipCode;
}

public void setStreet(String street) {
    this.street = street;
}

public void setCity(String city) {
    this.city = city;
}

public void setState(String state) {
    this.state = state;
}

public void setZipCode(String zipCode) {
    this.zipCode = zipCode;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Address address = (Address) o;
    return Objects.equals(street, address.street) && Objects.equals(city,
address.city) && Objects.equals(state, address.state) && Objects.equals(zipCode,
address.zipCode);
}

@Override
public int hashCode() {
    return Objects.hash(street, city, state, zipCode);
}

@Override
public String toString() {
    return "Address{" +
        "street='" + street + '\'' +
```

```
        ", city='" + city + '\'' +
        ", state='" + state + '\'' +
        ", zipCode='" + zipCode + '\'' +
        '}'};
    }
}
```

## PhoneNumber-Klasse

```
@DynamoDbBean
public class PhoneNumber {
    String type;
    String number;

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public String getNumber() {
        return number;
    }

    public void setNumber(String number) {
        this.number = number;
    }

    @Override
    public String toString() {
        return "PhoneNumber{" +
            "type='" + type + '\'' +
            ", number='" + number + '\'' +
            '}'};
    }
}
```

## Ordnen Sie verschachtelte Attribute zu

### Verwenden Sie kommentierte Klassen

Sie können verschachtelte Attribute für benutzerdefinierte Klassen speichern, indem Sie sie mit Anmerkungen versehen. Die `Address` Klasse und die `PhoneNumber` Klasse, die zuvor gezeigt wurden, sind nur mit der Anmerkung `@DynamoDbBean` annotiert. Wenn die DynamoDB Enhanced Client-API das Tabellenschema für die `Person` Klasse mit dem folgenden Codeausschnitt erstellt, erkennt die API die Verwendung der `PhoneNumber` Klassen `Address` und und erstellt die entsprechenden Mappings für die Arbeit mit DynamoDB.

```
TableSchema<Person> personTableSchema = TableSchema.fromBean(Person.class);
```

### Verwenden Sie verschachtelte Schemas

Der alternative Ansatz besteht darin, Schema-Builder für statische Tabellen für jede der Klassen zu verwenden, wie im folgenden Code gezeigt.

Die Tabellenschemas für die `PhoneNumber` Klassen `Address` und sind abstrakt in dem Sinne, dass sie nicht mit einer DynamoDB-Tabelle verwendet werden können. Das liegt daran, dass ihnen Definitionen für den Primärschlüssel fehlen. Sie werden jedoch als verschachtelte Schemas im Tabellenschema für die `Person` Klasse verwendet.

Nach den Kommentarzeilen 1 und 2 in der Definition von `Person` sehen Sie den Code `PERSON_TABLE_SCHEMA`, der die abstrakten Tabellenschemas verwendet. Die Verwendung von `documentOf` in der `EnhanceType.documentOf(...)` Methode bedeutet nicht, dass die Methode einen `EnhancedDocument` Typ der Enhanced Document API zurückgibt. Die `documentOf(...)` Methode gibt in diesem Kontext ein Objekt zurück, das weiß, wie es sein Klassenargument mithilfe des Tabellenschema-Arguments DynamoDB-Tabellenattributen zuordnen kann.

### Statischer Schemacode

```
// Abstract table schema that cannot be used to work with a DynamoDB table,  
// but can be used as a nested schema.  
public static final TableSchema<Address> TABLE_SCHEMA_ADDRESS =  
TableSchema.builder(Address.class)  
    .newItemSupplier(Address::new)  
    .addAttribute(String.class, a -> a.name("street")  
        .getter(Address::getStreet)  
        .setter(Address::setStreet))
```

```
.addAttribute(String.class, a -> a.name("city")
    .getter(Address::getCity)
    .setter(Address::setCity))
.addAttribute(String.class, a -> a.name("zipcode")
    .getter(Address::getZipCode)
    .setter(Address::setZipCode))
.addAttribute(String.class, a -> a.name("state")
    .getter(Address::getState)
    .setter(Address::setState))
.build();

// Abstract table schema that cannot be used to work with a DynamoDB table,
// but can be used as a nested schema.
public static final TableSchema<PhoneNumber> TABLE_SCHEMA_PHONENUMBER =
TableSchema.builder(PhoneNumber.class)
    .newItemSupplier(PhoneNumber::new)
    .addAttribute(String.class, a -> a.name("type")
        .getter(PhoneNumber::getType)
        .setter(PhoneNumber::setType))
    .addAttribute(String.class, a -> a.name("number")
        .getter(PhoneNumber::getNumber)
        .setter(PhoneNumber::setNumber))
    .build();

// A static table schema that can be used with a DynamoDB table.
// The table schema contains two nested schemas that are used to perform mapping
to/from DynamoDB.
public static final TableSchema<Person> PERSON_TABLE_SCHEMA =
    TableSchema.builder(Person.class)
        .newItemSupplier(Person::new)
        .addAttribute(Integer.class, a -> a.name("id")
            .getter(Person::getId)
            .setter(Person::setId)
            .addTag(StaticAttributeTags.primaryPartitionKey()))
        .addAttribute(String.class, a -> a.name("firstName")
            .getter(Person::getFirstName)
            .setter(Person::setFirstName))
        .addAttribute(String.class, a -> a.name("lastName")
            .getter(Person::getLastName)
            .setter(Person::setLastName))
        .addAttribute(Integer.class, a -> a.name("age")
            .getter(Person::getAge)
            .setter(Person::setAge))
```

```

        .addAttribute(EnhancedType.listOf(String.class), a ->
a.name("hobbies")
                .getter(Person::getHobbies)
                .setter(Person::setHobbies))
        .addAttribute(EnhancedType.mapOf(
                EnhancedType.of(String.class),
                // 1. Use mapping functionality of the Address table
schema.
                EnhancedType.documentOf(Address.class,
TABLE_SCHEMA_ADDRESS)), a -> a.name("addresses")
                .getter(Person::getAddresses)
                .setter(Person::setAddresses))
        .addAttribute(EnhancedType.listOf(
                // 2. Use mapping functionality of the PhoneNumber table
schema.
                EnhancedType.documentOf(PhoneNumber.class,
TABLE_SCHEMA_PHONENUMBER)), a -> a.name("phoneNumbers")
                .getter(Person::getPhoneNumbers)
                .setter(Person::setPhoneNumbers))
        .build();

```

## Verschachtelte Attribute für das Projekt

Für `query()` und `scan()` Methoden können Sie angeben, welche Attribute in den Ergebnissen zurückgegeben werden sollen, indem Sie Methodenaufrufen wie `addNestedAttributeToProject()` und `attributesToProject()` verwenden. Die DynamoDB Enhanced Client API konvertiert die Parameter des Java-Methodenaufrufs in [Projektionsausdrücke](#), bevor die Anforderung gesendet wird.

Das folgende Beispiel füllt die Person Tabelle mit zwei Elementen und führt dann drei Scanvorgänge durch.

Beim ersten Scan werden alle Elemente in der Tabelle abgerufen, um die Ergebnisse mit den anderen Scanvorgängen zu vergleichen.

Der zweite Scan verwendet die [addNestedAttributeToProject\(\)](#) Builder-Methode, um nur den `street` Attributwert zurückzugeben.

Der dritte Scanvorgang verwendet die [attributesToProject\(\)](#) Builder-Methode, um die Daten für das Attribut der ersten Ebene zurückzugeben, `hobbies`. Der Attributtyp von `hobbies` ist eine Liste. Um auf einzelne Listenelemente zuzugreifen, führen Sie einen `get()` Vorgang in der Liste aus.

```
    personDynamoDbTable = getDynamoDbEnhancedClient().table("Person",
PERSON_TABLE_SCHEMA);
    PersonUtils.createPersonTable(personDynamoDbTable, getDynamoDbClient());
    // Use a utility class to add items to the Person table.
    List<Person> personList = PersonUtils.getItemsForCount(2);
    // This utility method performs a put against DynamoDB to save the instances in
the list argument.
    PersonUtils.putCollection(getDynamoDbEnhancedClient(), personList,
personDynamoDbTable);

    // The first scan logs all items in the table to compare to the results of the
subsequent scans.
    final PageIterable<Person> allItems = personDynamoDbTable.scan();
    allItems.items().forEach(p ->
        // 1. Log what is in the table.
        logger.info(p.toString()));

    // Scan for nested attributes.
    PageIterable<Person> streetScanResult = personDynamoDbTable.scan(b -> b
        // Use the 'addNestedAttributeToProject()' or
'addNestedAttributesToProject()' to access data nested in maps in DynamoDB.
        .addNestedAttributeToProject(
            NestedAttributeName.create("addresses", "work", "street")
        ));

    streetScanResult.items().forEach(p ->
        //2. Log the results of requesting nested attributes.
        logger.info(p.toString()));

    // Scan for a top-level list attribute.
    PageIterable<Person> phoneNumbersScanResult = personDynamoDbTable.scan(b -> b
        // Use the 'attributesToProject()' method to access first-level
attributes.
        .attributesToProject("hobbies"));

    phoneNumbersScanResult.items().forEach((p) -> {
        // 3. Log the results of the request for the 'hobbies' attribute.
        logger.info(p.toString());
        // To access an item in a list, first get the parent attribute, 'hobbies',
then access items in the list.
        String hobby = p.getHobbies().get(1);
        // 4. Log an item in the list.
        logger.info(hobby);
    });
}
```

```
});
```

```
// Logged results from comment line 1.
Person{id=2, firstName='first name 2', lastName='last name 2', age=11,
  addresses={work=Address{street='street 21', city='city 21', state='state 21',
  zipCode='33333'}, home=Address{street='street 2', city='city 2', state='state 2',
  zipCode='22222'}}, phoneNumbers=[PhoneNumber{type='home', number='222-222-2222'},
  PhoneNumber{type='work', number='333-333-3333'}], hobbies=[hobby 2, hobby 21]}
Person{id=1, firstName='first name 1', lastName='last name 1', age=11,
  addresses={work=Address{street='street 11', city='city 11', state='state 11',
  zipCode='22222'}, home=Address{street='street 1', city='city 1', state='state 1',
  zipCode='11111'}}, phoneNumbers=[PhoneNumber{type='home', number='111-111-1111'},
  PhoneNumber{type='work', number='222-222-2222'}], hobbies=[hobby 1, hobby 11]}

// Logged results from comment line 2.
Person{id=null, firstName='null', lastName='null', age=null,
  addresses={work=Address{street='street 21', city='null', state='null',
  zipCode='null'}}, phoneNumbers=null, hobbies=null}
Person{id=null, firstName='null', lastName='null', age=null,
  addresses={work=Address{street='street 11', city='null', state='null',
  zipCode='null'}}, phoneNumbers=null, hobbies=null}

// Logged results from comment lines 3 and 4.
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
  phoneNumbers=null, hobbies=[hobby 2, hobby 21]}
hobby 21
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
  phoneNumbers=null, hobbies=[hobby 1, hobby 11]}
hobby 11
```

### Note

Wenn die `attributesToProject()` Methode einer anderen Builder-Methode folgt, die Attribute hinzufügt, die Sie projizieren möchten, `attributesToProject()` ersetzt die mitgelieferte Liste der Attributnamen alle anderen Attributnamen.

Ein Scan, der mit der `ScanEnhancedRequest` Instanz im folgenden Codeausschnitt durchgeführt wird, gibt nur Hobbydaten zurück.

```
ScanEnhancedRequest lastOverwrites = ScanEnhancedRequest.builder()
    .addNestedAttributeToProject(
        NestedAttributeName.create("addresses", "work", "street"))
```

```

        .addAttributeToProject("firstName")
        // If the 'attributesToProject()' method follows other builder methods
        that add attributes for projection,
        // its list of attributes replace all previous attributes.
        .attributesToProject("hobbies")
        .build();
PageIterable<Person> hobbiesOnlyResult =
    personDynamoDbTable.scan(lastOverwrites);
hobbiesOnlyResult.items().forEach(p ->
    logger.info(p.toString()));

// Logged results.
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
    phoneNumbers=null, hobbies=[hobby 2, hobby 21]}
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
    phoneNumbers=null, hobbies=[hobby 1, hobby 11]}

```

Der folgende Codeausschnitt verwendet die Methode zuerst. `attributesToProject()` Bei dieser Reihenfolge werden alle anderen angeforderten Attribute beibehalten.

```

ScanEnhancedRequest attributesPreserved = ScanEnhancedRequest.builder()
    // Use 'attributesToProject()' first so that the method call does not
    replace all other attributes
    // that you want to project.
    .attributesToProject("firstName")
    .addNestedAttributeToProject(
        NestedAttributeName.create("addresses", "work", "street"))
    .addAttributeToProject("hobbies")
    .build();
PageIterable<Person> allAttributesResult =
    personDynamoDbTable.scan(attributesPreserved);
allAttributesResult.items().forEach(p ->
    logger.info(p.toString()));

// Logged results.
Person{id=null, firstName='first name 2', lastName='null', age=null,
    addresses={work=Address{street='street 21', city='null', state='null',
    zipCode='null'}}, phoneNumbers=null, hobbies=[hobby 2, hobby 21]}
Person{id=null, firstName='first name 1', lastName='null', age=null,
    addresses={work=Address{street='street 11', city='null', state='null',
    zipCode='null'}}, phoneNumbers=null, hobbies=[hobby 1, hobby 11]}

```



## Leere Objekte konservieren mit `@DynamoDbPreserveEmptyObject`

Wenn Sie eine Bean mit leeren Objekten in Amazon DynamoDB speichern und möchten, dass das SDK die leeren Objekte beim Abrufen neu erstellt, kommentieren Sie den Getter der inneren Bean mit `@DynamoDbPreserveEmptyObject`

Um zu veranschaulichen, wie die Anmerkung funktioniert, verwendet das Codebeispiel die folgenden beiden Beans.

### Beispiel Bohnen

Die folgende Datenklasse enthält zwei InnerBean Felder. Die Getter-Methode `getInnerBeanWithoutAnno()`, ist nicht mit annotiert.

`@DynamoDbPreserveEmptyObject` Die `getInnerBeanWithAnno()` Methode ist mit Anmerkungen versehen.

```
@DynamoDbBean
public class MyBean {

    private String id;
    private String name;
    private InnerBean innerBeanWithoutAnno;
    private InnerBean innerBeanWithAnno;

    @DynamoDbPartitionKey
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public InnerBean getInnerBeanWithoutAnno() { return innerBeanWithoutAnno; }
    public void setInnerBeanWithoutAnno(InnerBean innerBeanWithoutAnno)
    { this.innerBeanWithoutAnno = innerBeanWithoutAnno; }

    @DynamoDbPreserveEmptyObject
    public InnerBean getInnerBeanWithAnno() { return innerBeanWithAnno; }
    public void setInnerBeanWithAnno(InnerBean innerBeanWithAnno)
    { this.innerBeanWithAnno = innerBeanWithAnno; }

    @Override
    public String toString() {
        return new StringJoiner(", ", MyBean.class.getSimpleName() + "[", "]")
```

```

        .add("innerBeanWithoutAnno=" + innerBeanWithoutAnno)
        .add("innerBeanWithAnno=" + innerBeanWithAnno)
        .add("id='" + id + "'")
        .add("name='" + name + "'")
        .toString();
    }
}

```

Instanzen der folgenden InnerBean Klasse sind Felder von MyBean und werden im Beispielcode als leere Objekte initialisiert.

```

@DynamoDbBean
public class InnerBean {

    private String innerBeanField;

    public String getInnerBeanField() {
        return innerBeanField;
    }

    public void setInnerBeanField(String innerBeanField) {
        this.innerBeanField = innerBeanField;
    }

    @Override
    public String toString() {
        return "InnerBean{" +
            "innerBeanField='" + innerBeanField + '\'' +
            '}';
    }
}

```

Das folgende Codebeispiel speichert ein MyBean Objekt mit initialisierten inneren Beans in DynamoDB und ruft dann das Element ab. Die protokollierte Ausgabe zeigt, dass das innerBeanWithoutAnno nicht initialisiert wurde, sondern erstellt wurde. innerBeanWithAnno

```

public MyBean preserveEmptyObjectAnnoUsingGetItemExample(DynamoDbTable<MyBean>
myBeanTable) {
    // Save an item to DynamoDB.
    MyBean bean = new MyBean();
    bean.setId("1");
    bean.setInnerBeanWithoutAnno(new InnerBean()); // Instantiate the inner bean.
}

```

```

bean.setInnerBeanWithAnno(new InnerBean());        // Instantiate the inner bean.
myBeanTable.putItem(bean);

GetItemEnhancedRequest request = GetItemEnhancedRequest.builder()
    .key(Key.builder().partitionValue("1").build())
    .build();
MyBean myBean = myBeanTable.getItem(request);

logger.info(myBean.toString());
// Output 'MyBean[innerBeanWithoutAnno=null,
innerBeanWithAnno=InnerBean{innerBeanField='null'}, id='1', name='null']'.

return myBean;
}

```

### Alternatives statisches Schema

Sie können die folgende `StaticTableSchema` Version der Tabellenschemas anstelle der Anmerkungen auf den Beans verwenden.

```

public static TableSchema<MyBean> buildStaticSchemas() {

    StaticTableSchema<InnerBean> innerBeanStaticTableSchema =
        StaticTableSchema.builder(InnerBean.class)
            .newItemSupplier(InnerBean::new)
            .addAttribute(String.class, a -> a.name("innerBeanField")
                .getter(InnerBean::getInnerBeanField)
                .setter(InnerBean::setInnerBeanField))
            .build();

    return StaticTableSchema.builder(MyBean.class)
        .newItemSupplier(MyBean::new)
        .addAttribute(String.class, a -> a.name("id")
            .getter(MyBean::getId)
            .setter(MyBean::setId)
            .addTag(primaryPartitionKey()))
        .addAttribute(String.class, a -> a.name("name")
            .getter(MyBean::getName)
            .setter(MyBean::setName))
        .addAttribute(EnhancedType.documentOf(InnerBean.class,
            innerBeanStaticTableSchema),
            a -> a.name("innerBean1")
                .getter(MyBean::getInnerBeanWithoutAnno))

```

```

        .setter(MyBean::setInnerBeanWithoutAnno))
    .addAttribute(EnhancedType.documentOf(InnerBean.class,
        innerBeanStaticTableSchema,
        b -> b.preserveEmptyObject(true)),
        a -> a.name("innerBean2")
        .getter(MyBean::getInnerBeanWithAnno)
        .setter(MyBean::setInnerBeanWithAnno))
    .build();
}

```

## Vermeiden Sie das Speichern von Null-Attributen verschachtelter Objekte

Sie können Null-Attribute verschachtelter Objekte überspringen, wenn Sie ein Datenklassenobjekt in DynamoDB speichern, indem Sie die Anmerkung anwenden. `@DynamoDbIgnoreNulls` Im Gegensatz dazu werden Attribute der obersten Ebene mit Nullwerten niemals in der Datenbank gespeichert.

Um zu veranschaulichen, wie die Anmerkung funktioniert, verwendet das Codebeispiel die folgenden beiden Beans.

### Beispiel Bohnen

Die folgende Datenklasse enthält zwei `InnerBean` Felder. Die Getter-Methode `getInnerBeanWithoutAnno()`, ist nicht annotiert. Die `getInnerBeanWithIgnoreNullsAnno()` Methode ist mit `@DynamoDbIgnoreNulls`

```

@DynamoDbBean
public class MyBean {

    private String id;
    private String name;
    private InnerBean innerBeanWithoutAnno;
    private InnerBean innerBeanWithIgnoreNullsAnno;

    @DynamoDbPartitionKey
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public InnerBean getInnerBeanWithoutAnno() { return innerBeanWithoutAnno; }
}

```

```

    public void setInnerBeanWithoutAnno(InnerBean innerBeanWithoutAnno)
    { this.innerBeanWithoutAnno = innerBeanWithoutAnno; }

    @DynamoDbIgnoreNulls
    public InnerBean getInnerBeanWithIgnoreNullsAnno() { return
innerBeanWithIgnoreNullsAnno; }
    public void setInnerBeanWithIgnoreNullsAnno(InnerBean innerBeanWithAnno)
    { this.innerBeanWithIgnoreNullsAnno = innerBeanWithAnno; }

    @Override
    public String toString() {
        return new StringJoiner(", ", MyBean.class.getSimpleName() + "[", "]")
            .add("innerBeanWithoutAnno=" + innerBeanWithoutAnno)
            .add("innerBeanWithIgnoreNullsAnno=" + innerBeanWithIgnoreNullsAnno)
            .add("id='" + id + "'")
            .add("name='" + name + "'")
            .toString();
    }
}

```

Instanzen der folgenden InnerBean Klasse sind Felder von MyBean und werden im folgenden Beispielcode verwendet.

```

@DynamoDbBean
public class InnerBean {

    private String innerBeanFieldString;
    private Integer innerBeanFieldInteger;

    public String getInnerBeanFieldString() { return innerBeanFieldString; }
    public void setInnerBeanFieldString(String innerBeanFieldString)
    { this.innerBeanFieldString = innerBeanFieldString; }

    public Integer getInnerBeanFieldInteger() { return innerBeanFieldInteger; }
    public void setInnerBeanFieldInteger(Integer innerBeanFieldInteger)
    { this.innerBeanFieldInteger = innerBeanFieldInteger; }

    @Override
    public String toString() {
        return new StringJoiner(", ", InnerBean.class.getSimpleName() + "[", "]")
            .add("innerBeanFieldString='" + innerBeanFieldString + "'")
            .add("innerBeanFieldInteger=" + innerBeanFieldInteger)
            .toString();
    }
}

```

```

    }
}

```

Im folgenden Codebeispiel wird ein `InnerBean` Objekt erstellt und nur einem seiner beiden Attribute ein Wert zugewiesen.

```

public void ignoreNullsAnnoUsingPutItemExample(DynamoDbTable<MyBean> myBeanTable) {
    // Create an InnerBean object and give only one attribute a value.
    InnerBean innerBeanOneAttributeSet = new InnerBean();
    innerBeanOneAttributeSet.setInnerBeanFieldInteger(200);

    // Create a MyBean instance and use the same InnerBean instance both for
attributes.
    MyBean bean = new MyBean();
    bean.setId("1");
    bean.setInnerBeanWithoutAnno(innerBeanOneAttributeSet);
    bean.setInnerBeanWithIgnoreNullsAnno(innerBeanOneAttributeSet);

    Map<String, AttributeValue> itemMap = myBeanTable.tableSchema().itemToMap(bean,
true);
    logger.info(itemMap.toString());
    // Log the map that is sent to the database.
    //
{innerBeanWithIgnoreNullsAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200)}),
id=AttributeValue(S=1),
innerBeanWithoutAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200),
innerBeanFieldString=AttributeValue(NUL=true)}})}

    // Save the MyBean object to the table.
    myBeanTable.putItem(bean);
}

```

Um die Low-Level-Daten zu visualisieren, die an DynamoDB gesendet werden, protokolliert der Code die Attributzuordnung, bevor das Objekt gespeichert wird. `MyBean`

Die protokollierte Ausgabe zeigt, dass ein Attribut `innerBeanWithIgnoreNullsAnno` ausgegeben wird,

```
innerBeanWithIgnoreNullsAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200)})
```

Die `innerBeanWithoutAnno` Instanz gibt zwei Attribute aus. Ein Attribut hat einen Wert von 200 und das andere ist ein Nullwertattribut.

```
innerBeanWithoutAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200),
innerBeanFieldString=AttributeValue(NUL=true)})
```

## JSON-Darstellung der Attributzuordnung

Die folgende JSON-Darstellung macht es einfacher, die in DynamoDB gespeicherten Daten zu sehen.

```
{
  "id": {
    "S": "1"
  },
  "innerBeanWithIgnoreNullsAnno": {
    "M": {
      "innerBeanFieldInteger": {
        "N": "200"
      }
    }
  },
  "innerBeanWithoutAnno": {
    "M": {
      "innerBeanFieldInteger": {
        "N": "200"
      },
      "innerBeanFieldString": {
        "NULL": true
      }
    }
  }
}
```

## Alternatives statisches Schema

Sie können die folgende `StaticTableSchema` Version der Tabellenschemas anstelle von Datenklassenanmerkungen verwenden.

```
public static TableSchema<MyBean> buildStaticSchemas() {

    StaticTableSchema<InnerBean> innerBeanStaticTableSchema =
        StaticTableSchema.builder(InnerBean.class)
            .newItemSupplier(InnerBean::new)
}
```

```
        .addAttribute(String.class, a -> a.name("innerBeanFieldString")
            .getter(InnerBean::getInnerBeanFieldString)
            .setter(InnerBean::setInnerBeanFieldString))
        .addAttribute(Integer.class, a -> a.name("innerBeanFieldInteger")
            .getter(InnerBean::getInnerBeanFieldInteger)
            .setter(InnerBean::setInnerBeanFieldInteger))
        .build();

return StaticTableSchema.builder(MyBean.class)
    .newItemSupplier(MyBean::new)
    .addAttribute(String.class, a -> a.name("id")
        .getter(MyBean::getId)
        .setter(MyBean::setId)
        .addTag(primaryPartitionKey()))
    .addAttribute(String.class, a -> a.name("name")
        .getter(MyBean::getName)
        .setter(MyBean::setName))
    .addAttribute(EnhancedType.documentOf(InnerBean.class,
        innerBeanStaticTableSchema),
        a -> a.name("innerBeanWithoutAnno")
            .getter(MyBean::getInnerBeanWithoutAnno)
            .setter(MyBean::setInnerBeanWithoutAnno))
    .addAttribute(EnhancedType.documentOf(InnerBean.class,
        innerBeanStaticTableSchema,
        b -> b.ignoreNulls(true)),
        a -> a.name("innerBeanWithIgnoreNullsAnno")
            .getter(MyBean::getInnerBeanWithIgnoreNullsAnno)
            .setter(MyBean::setInnerBeanWithIgnoreNullsAnno))
    .build();
}
```

## Arbeiten Sie mit JSON-Dokumenten mit der Enhanced Document API für DynamoDB

Die [Enhanced Document API](#) für AWS SDK for Java 2.x ist für die Arbeit mit dokumentenorientierten Daten konzipiert, die kein festes Schema haben. Sie können damit jedoch auch benutzerdefinierte Klassen verwenden, um einzelne Attribute zuzuordnen.

Die Enhanced Document API ist der Nachfolger der [Document API](#) der AWS SDK for Java Version v1.x.

### Inhalt

- [Beginnen Sie mit der Verwendung der Enhanced Document API](#)



- [Erstellen Sie ein und ein DocumentTableSchemaDynamoDbTable](#)
- [Erstellen Sie erweiterte Dokumente](#)
  - [Aus einer JSON-Zeichenfolge erstellen](#)
  - [Aus einzelnen Elementen zusammensetzen](#)
- [Führen Sie CRUD-Operationen durch](#)
- [Greifen Sie als benutzerdefinierte Objekte auf erweiterte Dokumentattribute zu](#)
- [Verwenden Sie eine EnhancedDocument ohne DynamoDB](#)

Beginnen Sie mit der Verwendung der Enhanced Document API

Die Enhanced Document API erfordert dieselben [Abhängigkeiten](#) wie die DynamoDB Enhanced Client API. Außerdem ist eine [DynamoDbEnhancedClientInstanz](#) erforderlich, wie am Anfang dieses Themas beschrieben.

Da die Enhanced Document API mit Version 2.20.3 von veröffentlicht wurde AWS SDK for Java 2.x, benötigen Sie diese Version oder eine höhere Version.

Erstellen Sie ein und ein **DocumentTableSchemaDynamoDbTable**

[Um mithilfe der Enhanced Document API Befehle für eine DynamoDB-Tabelle aufzurufen, verknüpfen Sie die Tabelle mit einem clientseitigen DynamoDbTable < > -Ressourcenobjekt. EnhancedDocument](#)

Die `table()` Methode des erweiterten Clients erstellt eine `DynamoDbTable<EnhancedDocument>` Instanz und benötigt Parameter für den DynamoDB-Tabellennamen und a. `DocumentTableSchema`

Der Builder für a [DocumentTableSchema](#) erfordert einen primären Indexschlüssel und einen oder mehrere Attributkonverter-Anbieter. Die `AttributeConverterProvider.defaultProvider()` Methode stellt Konverter für [Standardtypen](#) bereit. Sie sollte auch dann angegeben werden, wenn Sie einen benutzerdefinierten Attributkonverter-Anbieter angeben. Sie können dem Builder einen optionalen sekundären Indexschlüssel hinzufügen.

Der folgende Codeausschnitt zeigt den Code, der die clientseitige Darstellung einer DynamoDB-Tabelle generiert, in der schemalose Objekte gespeichert person werden. `EnhancedDocument`

```
DynamoDbTable<EnhancedDocument> documentDynamoDbTable =
    enhancedClient.table("person",
        TableSchema.documentSchemaBuilder()
            // Specify the primary key attributes.
```

```
.addIndexPartitionKey(TableMetadata.primaryIndexName(),"id", AttributeValueType.S)
    .addIndexSortKey(TableMetadata.primaryIndexName(),
"lastName", AttributeValueType.S)
    // Specify attribute converter providers. Minimally add the
default one.

.attributeConverterProviders(AttributeConverterProvider.defaultProvider())
    .build());

// Call documentTable.createTable() if "person" does not exist in DynamoDB.
// createTable() should be called only one time.
```

Im Folgenden wird die JSON-Darstellung eines Objekts gezeigt, die in diesem Abschnitt verwendet wirdperson.

### personJSON-Objekt

```
{
  "id": 1,
  "firstName": "Richard",
  "lastName": "Roe",
  "age": 25,
  "addresses":
  {
    "home": {
      "zipCode": "00000",
      "city": "Any Town",
      "state": "FL",
      "street": "123 Any Street"
    },
    "work": {
      "zipCode": "00001",
      "city": "Anywhere",
      "state": "FL",
      "street": "100 Main Street"
    }
  },
  "hobbies": [
    "Hobby 1",
    "Hobby 2"
  ],
  "phoneNumbers": [
```

```
{
  "type": "Home",
  "number": "555-0100"
},
{
  "type": "Work",
  "number": "555-0119"
}
]
```

## Erstellen Sie erweiterte Dokumente

An [EnhancedDocument](#) steht für ein Objekt vom Typ Dokument mit einer komplexen Struktur mit verschachtelten Attributen. An `EnhancedDocument` erfordert Attribute der obersten Ebene, die den für den angegebenen Primärschlüsselattributen entsprechen. `DocumentTableSchema` Der restliche Inhalt ist willkürlich und kann aus Attributen der obersten Ebene und auch tief verschachtelten Attributen bestehen.

Sie erstellen eine `EnhancedDocument` Instanz mithilfe eines Builders, der mehrere Möglichkeiten zum Hinzufügen von Elementen bietet.

### Aus einer JSON-Zeichenfolge erstellen

Mit einer JSON-Zeichenfolge können Sie einen Methodenaufruf `EnhancedDocument` in einem Schritt erstellen. Das folgende Snippet erstellt `EnhancedDocument` aus einer JSON-Zeichenfolge, die von der `jsonPerson()` Hilfsmethode zurückgegeben wird. Die `jsonPerson()` Methode gibt die JSON-String-Version des zuvor gezeigten [Personenobjekts](#) zurück.

```
EnhancedDocument document =
    EnhancedDocument.builder()
        .json( jsonPerson() )
        .build();
```

### Aus einzelnen Elementen zusammensetzen

Alternativ können Sie mit den typsicheren Methoden des Builders eine `EnhancedDocument` Instanz aus einzelnen Komponenten erstellen.

Im folgenden Beispiel wird ein `person` erweitertes Dokument erstellt, das dem erweiterten Dokument ähnelt, das aus der JSON-Zeichenfolge im vorherigen Beispiel erstellt wurde.

/\* Define the shape of an address map whose JSON representation looks like the following.

Use 'addressMapEnhancedType' in the following EnhancedDocument.builder() to simplify the code.

```
"home": {
  "zipCode": "00000",
  "city": "Any Town",
  "state": "FL",
  "street": "123 Any Street"
}*/
```

```
EnhancedType<Map<String, String>> addressMapEnhancedType =
    EnhancedType.mapOf(EnhancedType.of(String.class),
EnhancedType.of(String.class));
```

// Use the builder's typesafe methods to add elements to the enhanced document.

```
EnhancedDocument personDocument = EnhancedDocument.builder()
    .putNumber("id", 50)
    .putString("firstName", "Shirley")
    .putString("lastName", "Rodriguez")
    .putNumber("age", 53)
    .putNull("nullAttribute")
    .putJson("phoneNumbers", phoneNumbersJSONString())
```

/\* Add the map of addresses whose JSON representation looks like the following.

```
{
  "home": {
    "zipCode": "00000",
    "city": "Any Town",
    "state": "FL",
    "street": "123 Any Street"
  }
} */
    .putMap("addresses", getAddresses(), EnhancedType.of(String.class),
addressMapEnhancedType)
    .putList("hobbies", List.of("Theater", "Golf"),
EnhancedType.of(String.class))
    .build();
```

## Hilfsmethoden

```
private static String phoneNumbersJSONString() {
```

```

    return " [" +
        " {" +
        "   \"type\": \"Home\"," +
        "   \"number\": \"555-0140\"" +
        " }," +
        " {" +
        "   \"type\": \"Work\"," +
        "   \"number\": \"555-0155\"" +
        " }" +
        " ]";
}

private static Map<String, Map<String, String>> getAddresses() {
    return Map.of(
        "home", Map.of(
            "zipCode", "00002",
            "city", "Any Town",
            "state", "ME",
            "street", "123 Any Street"));
}

```

Führen Sie CRUD-Operationen durch

Nachdem Sie eine EnhancedDocument Instanz definiert haben, können Sie sie in einer DynamoDB-Tabelle speichern. Der folgende Codeausschnitt verwendet das [PersonDocument, das aus einzelnen Elementen](#) erstellt wurde.

```
documentDynamoDbTable.putItem(personDocument);
```

Nachdem Sie eine erweiterte Dokumentinstanz aus DynamoDB gelesen haben, können Sie die einzelnen Attributwerte mithilfe von Getter extrahieren, wie im folgenden Codeausschnitt gezeigt, die auf die Daten zugreifen, die aus dem gespeichert wurden. personDocument Alternativ können Sie den gesamten Inhalt in eine JSON-Zeichenfolge extrahieren, wie im letzten Teil des Beispielcodes gezeigt.

```

// Read the item.
EnhancedDocument personDocFromDb =
documentDynamoDbTable.getItem(Key.builder().partitionValue(50).build());

// Access top-level attributes.

```

```
logger.info("Name: {} {}", personDocFromDb.getString("firstName"),
personDocFromDb.getString("lastName"));
// Name: Shirley Rodriguez

// Typesafe access of a deeply nested attribute. The addressMapEnhancedType
shown previously defines the shape of an addresses map.
Map<String, Map<String, String>> addresses =
personDocFromDb.getMap("addresses", EnhancedType.of(String.class),
addressMapEnhancedType);
addresses.keySet().forEach(k -> logger.info(addresses.get(k).toString()));
// {zipCode=00002, city=Any Town, street=123 Any Street, state=ME}

// Alternatively, work with AttributeValue types checking along the way for
deeply nested attributes.
Map<String, AttributeValue> addressesMap =
personDocFromDb.getMapOfUnknownType("addresses");
addressesMap.keySet().forEach((String k) -> {
    logger.info("Looking at data for [{}] address", k);
    // Looking at data for [home] address
    AttributeValue value = addressesMap.get(k);
    AttributeValue cityValue = value.m().get("city");
    if (cityValue != null) {
        logger.info(cityValue.s());
        // Any Town
    }
});

List<AttributeValue> phoneNumbers =
personDocFromDb.getListOfUnknownType("phoneNumbers");
phoneNumbers.forEach((AttributeValue av) -> {
    if (av.hasM()) {
        AttributeValue type = av.m().get("type");
        if (type.s() != null) {
            logger.info("Type of phone: {}", type.s());
            // Type of phone: Home
            // Type of phone: Work
        }
    }
});

String jsonPerson = personDocFromDb.toJson();
logger.info(jsonPerson);
```

```
// {"firstName":"Shirley","lastName":"Rodriguez","addresses":
{"home":{"zipCode":"00002","city":"Any Town","street":"123 Any
Street","state":"ME"}}, "hobbies":["Theater","Golf"],
//      "id":50,"nullAttribute":null,"age":53,"phoneNumbers":
[{"number":"555-0140","type":"Home"}, {"number":"555-0155","type":"Work"}]}
```

EnhancedDocument Instanzen können mit jeder Methode von [DynamoDbTable](#) oder [DynamoDbEnhancedClient](#) anstelle von zugewiesenen Datenklassen verwendet werden.

Greifen Sie als benutzerdefinierte Objekte auf erweiterte Dokumentattribute zu

Die Enhanced Document API bietet nicht nur eine API zum Lesen und Schreiben von Attributen mit schemalosen Strukturen, sondern ermöglicht auch die Konvertierung von Attributen in und aus Instanzen benutzerdefinierter Klassen.

Die Enhanced Document API verwendet `AttributeConverterProviders` und `AttributeConverters`, die im Abschnitt zur [Konvertierung von Steuerattributen](#) als Teil der DynamoDB Enhanced Client API angezeigt wurden.

Im folgenden Beispiel verwenden wir ein `CustomAttributeConverterProvider` mit seiner verschachtelten `AddressConverter` Klasse, um Objekte zu konvertieren. `Address`

Dieses Beispiel zeigt, dass Sie Daten aus Klassen und auch Daten aus Strukturen kombinieren können, die nach Bedarf erstellt werden. Dieses Beispiel zeigt auch, dass benutzerdefinierte Klassen auf jeder Ebene einer verschachtelten Struktur verwendet werden können. Die `Address` Objekte in diesem Beispiel sind Werte, die in einer Map verwendet werden.

```
public static void attributeToAddressClassMappingExample(DynamoDbEnhancedClient
enhancedClient, DynamoDbClient standardClient) {
    String tableName = "customer";

    // Define the DynamoDbTable for an enhanced document.
    // The schema builder provides methods for attribute converter providers and
    keys.
    DynamoDbTable<EnhancedDocument> documentDynamoDbTable =
enhancedClient.table(tableName,
        DocumentTableSchema.builder()
            // Add the CustomAttributeConverterProvider along with the
            default when you build the table schema.
            .attributeConverterProviders(
                List.of(
                    new CustomAttributeConverterProvider(),
```

```

        AttributeConverterProvider.defaultProvider()))
        .addIndexPartitionKey(TableMetadata.primaryIndexName(), "id",
AttributeValueType.N)
        .addIndexSortKey(TableMetadata.primaryIndexName(), "lastName",
AttributeValueType.S)
        .build());
// Create the DynamoDB table if needed.
documentDynamoDbTable.createTable();
waitForTableCreation(tableName, standardClient);

// The getAddressessForCustomMappingExample() helper method that provides
'addresses' shows the use of a custom Address class
// rather than using a Map<String, Map<String, String> to hold the address
data.
Map<String, Address> addresses = getAddressessForCustomMappingExample();

// Build an EnhancedDocument instance to save an item with a mix of structures
defined as needed and static classes.
EnhancedDocument personDocument = EnhancedDocument.builder()
    .putNumber("id", 50)
    .putString("firstName", "Shirley")
    .putString("lastName", "Rodriguez")
    .putNumber("age", 53)
    .putNull("nullAttribute")
    .putJson("phoneNumbers", phoneNumbersJSONString())
    // Note the use of 'EnhancedType.of(Address.class)' instead of the more
generic
    // 'EnhancedType.mapOf(EnhancedType.of(String.class),
EnhancedType.of(String.class))' that was used in a previous example.
    .putMap("addresses", addresses, EnhancedType.of(String.class),
EnhancedType.of(Address.class))
    .putList("hobbies", List.of("Hobby 1", "Hobby 2"),
EnhancedType.of(String.class))
    .build();
// Save the item to DynamoDB.
documentDynamoDbTable.putItem(personDocument);

// Retrieve the item just saved.
EnhancedDocument srPerson =
documentDynamoDbTable.getItem(Key.builder().partitionValue(50).sortValue("Rodriguez").build())

// Access the addresses attribute.
Map<String, Address> srAddresses = srPerson.get("addresses",

```



```

        EnhancedType.mapOf(EnhancedType.of(String.class),
        EnhancedType.of(Address.class)));

        srAddresses.keySet().forEach(k -> logger.info(addresses.get(k).toString()));

        documentDynamoDbTable.deleteTable();

// The content logged to the console shows that the saved maps were converted to
// Address instances.
Address{street='123 Main Street', city='Any Town', state='NC', zipCode='00000'}
Address{street='100 Any Street', city='Any Town', state='NC', zipCode='00000'}

```

## CustomAttributeConverterProviderCode

```

public class CustomAttributeConverterProvider implements AttributeConverterProvider {

    private final Map<EnhancedType<?>, AttributeConverter<?>> converterCache =
    ImmutableMap.of(
        // 1. Add AddressConverter to the internal cache.
        EnhancedType.of(Address.class), new AddressConverter());

    public static CustomAttributeConverterProvider create() {
        return new CustomAttributeConverterProvider();
    }

    // 2. The enhanced client queries the provider for attribute converters if it
    // encounters a type that it does not know how to convert.
    @SuppressWarnings("unchecked")
    @Override
    public <T> AttributeConverter<T> converterFor(EnhancedType<T> enhancedType) {
        return (AttributeConverter<T>) converterCache.get(enhancedType);
    }

    // 3. Custom attribute converter
    private class AddressConverter implements AttributeConverter<Address> {
        // 4. Transform an Address object into a DynamoDB map.
        @Override
        public AttributeValue transformFrom(Address address) {

            Map<String, AttributeValue> attributeValueMap = Map.of(
                "street", AttributeValue.fromS(address.getStreet()),
                "city", AttributeValue.fromS(address.getCity()),
                "state", AttributeValue.fromS(address.getState()),

```

```

        "zipCode", AttributeValue.fromS(address.getZipCode()));

    return AttributeValue.fromM(attributeValueMap);
}

// 5. Transform the DynamoDB map attribute to an Address object.
@Override
public Address transformTo(AttributeValue attributeValue) {
    Map<String, AttributeValue> m = attributeValue.m();
    Address address = new Address();
    address.setStreet(m.get("street").s());
    address.setCity(m.get("city").s());
    address.setState(m.get("state").s());
    address.setZipCode(m.get("zipCode").s());

    return address;
}

@Override
public EnhancedType<Address> type() {
    return EnhancedType.of(Address.class);
}

@Override
public AttributeValueType attributeValueType() {
    return AttributeValueType.M;
}
}
}

```

## Address-Klasse

```

public class Address {
    private String street;
    private String city;
    private String state;
    private String zipCode;

    public Address() {
    }

    public String getStreet() {
    return this.street;
    }
}

```

```
    }

    public String getCity() {
        return this.city;
    }

    public String getState() {
        return this.state;
    }

    public String getZipCode() {
        return this.zipCode;
    }

    public void setStreet(String street) {
        this.street = street;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public void setState(String state) {
        this.state = state;
    }

    public void setZipCode(String zipCode) {
        this.zipCode = zipCode;
    }
}
```

Hilfsmethode, die Adressen bereitstellt

Die folgende Hilfsmethode stellt die Map bereit, die benutzerdefinierte Address Instanzen für Werte anstelle von generischen Map<String, String> Instanzen für Werte verwendet.

```
private static Map<String, Address> getAddressesForCustomMappingExample() {
    Address homeAddress = new Address();
    homeAddress.setStreet("100 Any Street");
    homeAddress.setCity("Any Town");
    homeAddress.setState("NC");
    homeAddress.setZipCode("00000");
}
```

```

    Address workAddress = new Address();
    workAddress.setStreet("123 Main Street");
    workAddress.setCity("Any Town");
    workAddress.setState("NC");
    workAddress.setZipCode("00000");

    return Map.of("home", homeAddress,
                 "work", workAddress);
}

```

## Verwenden Sie eine **EnhancedDocument** ohne DynamoDB

Normalerweise verwenden Sie eine Instanz von `EnhancedDocument` um DynamoDB-Elemente vom Typ Dokument zu lesen und zu schreiben, sie kann jedoch auch unabhängig von DynamoDB verwendet werden.

Sie können ihre Fähigkeit verwenden `EnhancedDocuments`, zwischen JSON-Zeichenfolgen oder benutzerdefinierten Objekten in Low-Level-Maps von `AttributeValues` zu konvertieren, wie im folgenden Beispiel gezeigt.

```

public static void conversionWithoutDynamoDbExample() {
    Address address = new Address();
    address.setCity("my city");
    address.setState("my state");
    address.setStreet("my street");
    address.setZipCode("00000");

    // Build an EnhancedDocument instance for its conversion functionality alone.
    EnhancedDocument addressEnhancedDoc = EnhancedDocument.builder()
        // Important: You must specify attribute converter providers when you
    build an EnhancedDocument instance not used with a DynamoDB table.
        .attributeConverterProviders(new CustomAttributeConverterProvider(),
    DefaultAttributeConverterProvider.create())
        .put("addressDoc", address, Address.class)
        .build();

    // Convert address to a low-level item representation.
    final Map<String, AttributeValue> addressAsAttributeMap =
    addressEnhancedDoc.getMapOfUnknownType("addressDoc");
    logger.info("addressAsAttributeMap: {}", addressAsAttributeMap.toString());

    // Convert address to a JSON string.
    String addressAsJsonString = addressEnhancedDoc.toJson("addressDoc");
}

```

```
logger.info("addressAsJsonString: {}", addressAsJsonString);
// Convert addressEnhancedDoc back to an Address instance.
Address addressConverted = addressEnhancedDoc.get("addressDoc",
Address.class);
logger.info("addressConverted: {}", addressConverted.toString());
}

/* Console output:
    addressAsAttributeMap: {zipCode=AttributeValue(S=000000),
state=AttributeValue(S=my state), street=AttributeValue(S=my street),
city=AttributeValue(S=my city)}
    addressAsJsonString: {"zipCode":"000000","state":"my state","street":"my
street","city":"my city"}
    addressConverted: Address{street='my street', city='my city', state='my
state', zipCode='000000'}
*/
```

### Note

Wenn Sie ein erweitertes Dokument verwenden, das unabhängig von einer DynamoDB-Tabelle ist, stellen Sie sicher, dass Sie im Builder explizit Attributkonverter-Anbieter festlegen. Im Gegensatz dazu stellt das Dokumenttabellenschema die Konverteranbieter bereit, wenn ein erweitertes Dokument mit einer DynamoDB-Tabelle verwendet wird.

## Verwenden Sie Erweiterungen

Die DynamoDB Enhanced Client API unterstützt Plugin-Erweiterungen, die Funktionen bieten, die über Mapping-Operationen hinausgehen. Erweiterungen haben zwei Hook-Methoden, `undbeforeWrite()` und `afterRead()`. `beforeWrite()` ändert einen Schreibvorgang, bevor er stattfindet, und die `afterRead()` Methode ändert die Ergebnisse eines Lesevorgangs, nachdem er ausgeführt wurde. Da bei einigen Vorgängen (z. B. bei Elementaktualisierungen) sowohl ein Schreib- als auch ein Lesevorgang ausgeführt wird, werden beide Hook-Methoden aufgerufen.

Erweiterungen werden in der Reihenfolge geladen, in der sie im erweiterten Client Builder angegeben sind. Die Ladereihenfolge kann wichtig sein, da eine Erweiterung auf Werte reagieren kann, die durch eine vorherige Erweiterung transformiert wurden.

Die erweiterte Client-API enthält eine Reihe von Plugin-Erweiterungen, die sich im [extensions](#) Paket befinden. Standardmäßig lädt der erweiterte Client die [VersionedRecordExtension](#)

und die [AtomicCounterExtension](#). Sie können das Standardverhalten mit dem Enhance Client Builder überschreiben und jede Erweiterung laden. Sie können auch keine angeben, wenn Sie die Standarderweiterungen nicht verwenden möchten.

Wenn Sie Ihre eigenen Erweiterungen laden, lädt der erweiterte Client keine Standarderweiterungen. Wenn Sie das Verhalten einer der Standarderweiterungen verwenden möchten, müssen Sie sie explizit zur Liste der Erweiterungen hinzufügen.

Im folgenden Beispiel `verifyChecksumExtension` wird eine benutzerdefinierte Erweiterung mit dem Namen `VersionedRecordExtension`, die normalerweise standardmäßig von selbst geladen wird. Die `AtomicCounterExtension` ist in diesem Beispiel nicht geladen.

```
DynamoDbEnhancedClientExtension versionedRecordExtension =
    VersionedRecordExtension.builder().build();

DynamoDbEnhancedClient enhancedClient =
    DynamoDbEnhancedClient.builder()
        .dynamoDbClient(dynamoDbClient)
        .extensions(versionedRecordExtension,
            verifyChecksumExtension)
        .build();
```

## VersionedRecordExtension

Der `VersionedRecordExtension` wird standardmäßig geladen und erhöht die Versionsnummer eines Elements und verfolgt, wenn Elemente in die Datenbank geschrieben werden. Zu jedem Schreibvorgang wird eine Bedingung hinzugefügt, die dazu führt, dass der Schreibvorgang fehlschlägt, wenn die Versionsnummer des tatsächlich persistenten Elements nicht dem Wert entspricht, den die Anwendung zuletzt gelesen hat. Dieses Verhalten sorgt effektiv für optimistisches Sperren von Elementaktualisierungen. Wenn ein anderer Prozess ein Element zwischen dem Zeitpunkt aktualisiert, an dem der erste Prozess das Element gelesen hat und ein Update darauf geschrieben hat, schlägt der Schreibvorgang fehl.

Um anzugeben, welches Attribut verwendet werden soll, um die Versionsnummer des Elements nachzuverfolgen, kennzeichnen Sie ein numerisches Attribut im Tabellenschema.

Der folgende Ausschnitt gibt an, dass das `version` Attribut die Versionsnummer des Artikels enthalten soll.

```
@DynamoDbVersionAttribute
public Integer getVersion() {...};
```

```
public void setVersion(Integer version) {...};
```

Der entsprechende Ansatz für ein statisches Tabellenschema wird im folgenden Codeausschnitt dargestellt.

```
.addAttribute(Integer.class, a -> a.name("version")
    .getter(Customer::getVersion)
    .setter(Customer::setVersion)
    // Apply the 'version' tag to the attribute.

.tags(VersionedRecordExtension.AttributeTags.versionAttribute())

.tags(VersionedRecordExtension.AttributeTags.versionAttribute())
```

### AtomicCounterExtension

Das `AtomicCounterExtension` wird standardmäßig geladen und erhöht jedes Mal, wenn ein Datensatz in die Datenbank geschrieben wird, ein mit Tags versehenes numerisches Attribut. Start- und Inkrementwerte können angegeben werden. Wenn keine Werte angegeben sind, wird der Startwert auf 0 gesetzt und der Wert des Attributs wird um 1 erhöht.

Um anzugeben, welches Attribut ein Zähler ist, kennzeichnen Sie ein Attribut vom Typ `Long` im Tabellenschema.

Der folgende Ausschnitt zeigt die Verwendung der standardmäßigen Start- und Inkrementwerte für das Attribut `counter`

```
@DynamoDbAtomicCounter
public Long getCounter() {...};
public void setCounter(Long counter) {...};
```

Der Ansatz eines statischen Tabellenschemas wird im folgenden Codeausschnitt dargestellt. Die `Atomic Counter Extension` verwendet einen Startwert von 10 und erhöht den Wert jedes Mal, wenn der Datensatz geschrieben wird, um 5.

```
.addAttribute(Integer.class, a -> a.name("counter")
    .getter(Customer::getCounter)
    .setter(Customer::setCounter)
    // Apply the 'atomicCounter' tag to the
attribute with start and increment values.
    .tags(StaticAttributeTags.atomicCounter(10L,
5L))
```

## AutoGeneratedTimestampRecordExtension

Jedes Mal, wenn das Element erfolgreich in die Datenbank geschrieben wurde, werden die markierten Attribute des Typs `AutoGeneratedTimestampRecordExtension` automatisch [Instant](#) mit einem aktuellen Zeitstempel aktualisiert.

Diese Erweiterung ist standardmäßig nicht geladen. Daher müssen Sie sie als benutzerdefinierte Erweiterung angeben, wenn Sie den erweiterten Client erstellen, wie im ersten Beispiel in diesem Thema gezeigt.

Um anzugeben, welches Attribut mit dem aktuellen Zeitstempel aktualisiert werden soll, kennzeichnen Sie das `Instant` Attribut im Tabellenschema.

Das `lastUpdate` Attribut ist das Ziel des Erweiterungsverhaltens im folgenden Codeausschnitt. Beachten Sie die Anforderung, dass das Attribut ein `Instant` Typ sein muss.

```
@DynamoDbAutoGeneratedTimestampAttribute
public Instant getLastUpdate() {...}
public void setLastUpdate(Instant lastUpdate) {...}
```

Der entsprechende Ansatz für ein statisches Tabellenschema wird im folgenden Codeausschnitt dargestellt.

```
.addAttribute(Instant.class, a -> a.name("lastUpdate")
                .getter(Customer::getLastUpdate)
                .setter(Customer::setLastUpdate)
                // Applying the 'autoGeneratedTimestamp' tag to
the attribute.

.tags(AutoGeneratedTimestampRecordExtension.AttributeTags.autoGeneratedTimestampAttribute())
```

## Benutzerdefinierte Erweiterungen

Die folgende benutzerdefinierte Erweiterungsklasse zeigt eine `beforeWrite()` Methode, die einen Aktualisierungsausdruck verwendet. Nach Kommentarzeile 2 erstellen wir ein `SetAction` um das `registrationDate` Attribut festzulegen, falls das Element in der Datenbank noch kein `registrationDate` Attribut hat. Immer wenn ein `Customer` Objekt aktualisiert wird, stellt die Erweiterung sicher, dass `a` gesetzt `registrationDate` ist.

```
public final class CustomExtension implements DynamoDbEnhancedClientExtension {
```



```

// 1. In a custom extension, use an UpdateExpression to define what action to take
before
//   an item is updated.
@Override
public WriteModification beforeWrite(DynamoDbExtensionContext.BeforeWrite context)
{
    if ( context.operationContext().tableName().equals("Customer")
        && context.operationName().equals(OperationName.UPDATE_ITEM)) {
        return WriteModification.builder()
            .updateExpression(createUpdateExpression())
            .build();
    }
    return WriteModification.builder().build(); // Return an "empty"
WriteModification instance if the extension should not be applied.
// In this case, if the code is
not updating an item on the Customer table.
}

private static UpdateExpression createUpdateExpression() {

    // 2. Use a SetAction, a subclass of UpdateAction, to provide the values in the
update.
    SetAction setAction =
        SetAction.builder()
            .path("registrationDate")
            .value("if_not_exists(registrationDate, :regValue)")
            .putExpressionValue(":regValue",
AttributeValue.fromS(Instant.now().toString()))
            .build();
    // 3. Build the UpdateExpression with one or more UpdateAction.
    return UpdateExpression.builder()
        .addAction(setAction)
        .build();
}
}

```

## Verwenden Sie die DynamoDB Enhanced Client API asynchron

Wenn Ihre Anwendung nicht blockierende, asynchrone Aufrufe von DynamoDB erfordert, können Sie den verwenden. [DynamoDbEnhancedAsyncClient](#) Sie ähnelt der synchronen Implementierung, weist jedoch die folgenden wesentlichen Unterschiede auf:

1. Wenn Sie den `createDynamoDbEnhancedAsyncClient`, müssen Sie die asynchrone Version des Standardclients `createDynamoDbAsyncClient`, wie im folgenden Codeausschnitt dargestellt.

```
DynamoDbEnhancedAsyncClient enhancedClient =
    DynamoDbEnhancedAsyncClient.builder()
        .dynamoDbClient(dynamoDbAsyncClient)
        .build();
```

2. Methoden, die ein einzelnes Datenobjekt zurückgeben, geben nicht nur das Ergebnis zurück. `CompletableFuture` Ihre Anwendung kann dann andere Aufgaben ausführen, ohne das Ergebnis blockieren zu müssen. Der folgende Ausschnitt zeigt die asynchrone `getItem()` Methode.

```
CompletableFuture<Customer> result = customerDynamoDbTable.getItem(customer);
// Perform other work here.
return result.join(); // Now block and wait for the result.
```

3. Methoden, die paginierte Ergebnislisten zurückgeben, geben [SdkPublisher](#) statt eines zurück [SdkIterable](#), das die synchrone Methode für dieselben Methoden `DynamoDbEnhancedClient` zurückgibt. Ihre Anwendung kann dann einen Handler für diesen Herausgeber abonnieren, um die Ergebnisse asynchron zu verarbeiten, ohne sie blockieren zu müssen.

```
PagePublisher<Customer> results = customerDynamoDbTable.query(r ->
    r.queryConditional(keyEqualTo(k -> k.partitionValue("Smith"))));
results.subscribe(myCustomerResultsProcessor);
// Perform other work and let the processor handle the results asynchronously.
```

Ein vollständigeres Beispiel für die Arbeit mit dem `SdkPublisher` API finden Sie in [dem Beispiel](#) in dem Abschnitt, in dem die asynchrone `scan()` Methode beschrieben wird.

## Anmerkungen zu Datenklassen

Die folgende Tabelle listet die Anmerkungen auf, die für Datenklassen verwendet werden können, und enthält Links zu Informationen und Beispielen in diesem Handbuch. Die Tabelle wird in aufsteigender alphabetischer Reihenfolge nach Name der Anmerkung sortiert.

## In diesem Handbuch verwendete Datenklassenanmerkungen

Name der Anmerkung	Anmerkung gilt für*	Was es macht	Wo es in diesem Handbuch gezeigt wird
DynamoDbAtomicCounter	Attribut	Erhöht jedes Mal, wenn ein Datensatz in die Datenbank geschrieben wird, ein markiertes numerisches Attribut.	<a href="#">Einführung und Diskussion.</a>
DynamoDbAttribute	Attribut	Definiert oder benennt eine Bean-Eigenschaft um, die einem DynamoDB-Tabellenattribut zugeordnet ist.	<ul style="list-style-type: none"> <li>• <a href="#">Erste Diskussion.</a></li> <li>• <a href="#">Abschnitt „Erste Schritte“ – siehe Hinweis.</a></li> <li>• <a href="#">In MovieActor der Klasse In Beispiele für Abfragemethoden.</a></li> </ul>
DynamoDbAutoGeneratedTimestampAttribute	Attribut	Aktualisiert ein markiertes Attribut mit einem aktuellen Zeitstempel, wenn das Element erfolgreich in die Datenbank geschrieben wurde	<a href="#">Einführung und Diskussion.</a>
DynamoDbBean	class	Kennzeichnet eine Datenklasse als einem Tabellenschema zugeordnet.	Verwenden Sie zunächst in der <a href="#">Kundenklasse</a> im Abschnitt Erste Schritte. Im gesamten Handbuch werden

Name der Anmerkung	Anmerkung gilt für*	Was es macht	Wo es in diesem Handbuch gezeigt wird
			mehrere Verwendungen angezeigt.
DynamoDbConvertedBy	Attribut	Ordnet einen benutzerdefinierten Attribute Converter dem kommentierten Attribut zu.	<a href="#">Erste Diskussion und Beispiel.</a>
DynamoDbFlatten	Attribut	Verflacht alle Attribute einer separaten DynamoDB-Datenklasse und fügt sie als Attribute der obersten Ebene zum gelesenen und in die Datenbank geschriebenen Datensatz hinzu.	<ul style="list-style-type: none"> <li>• <a href="#">Erste Diskussion.</a></li> <li>• <a href="#">Auswirkungen auf anderen Code.</a></li> </ul>
DynamoDbIgnore	Attribut	Dies führt dazu, dass das Attribut nicht zugeordnet bleibt.	<ul style="list-style-type: none"> <li>• <a href="#">Erste Diskussion.</a></li> <li>• <a href="#">Verwenden Sie in der ProductCatalog-Klasse.</a></li> </ul>
DynamoDbIgnoreNulls	Attribut	Verhindert das Speichern von Nullattributen verschachtelter DynamoDb Objekte.	<a href="#">Diskussion und Beispiele.</a>

Name der Anmerkung	Anmerkung gilt für*	Was es macht	Wo es in diesem Handbuch gezeigt wird
DynamoDbImmutable	class	Kennzeichnet eine unveränderliche Datenklasse als einem Tabellenschema zugeordnet.	<ul style="list-style-type: none"> <li>• <a href="#">Einführung in die Anmerkung.</a></li> <li>• <a href="#">Verwenden Sie in der ProductCatalog-Klasse.</a></li> <li>• <a href="#">Verwenden Sie mit Lombok.</a></li> </ul>
DynamoDbPartitionKey	Attribut	Markiert ein Attribut als primären Partitionsschlüssel (Hash-Schlüssel) der DynamoDb Tabelle.	<ul style="list-style-type: none"> <li>• <a href="#">Erste Verwendung in der Kundenklasse im Abschnitt Erste Schritte.</a></li> <li>• <a href="#">Mit Lombok.</a></li> </ul>
DynamoDbP reserveEmptyObject	Attribut	Gibt an, dass das Objekt mit allen Nullfeldern initialisiert werden soll, wenn keine Daten für das Objekt vorhanden sind, das dem mit Anmerkungen versehenen Attribut zugeordnet ist.	<a href="#">Diskussion und Beispiele.</a>

Name der Anmerkung	Anmerkung gilt für*	Was es macht	Wo es in diesem Handbuch gezeigt wird
DynamoDbSecondaryPartitionKey	Attribut	Markiert ein Attribut als Partitionsschlüssel für einen globalen sekundären Index.	<ul style="list-style-type: none"> <li>• <a href="#">Verwenden Sie in sekundären Indizes und im Beispiel.</a></li> <li>• <a href="#">In den Beispielen für Abfragemethoden.</a></li> <li>• <a href="#">Im Lombok-Beispiel</a></li> <li>• <a href="#">Bei unveränderlichen Klassen.</a></li> </ul>
DynamoDbSecondarySortKey	Attribut	Markiert ein Attribut als optionalen Sortierschlüssel für einen globalen oder lokalen sekundären Index.	<ul style="list-style-type: none"> <li>• <a href="#">Verwenden Sie in sekundären Indizes und im Beispiel.</a></li> <li>• <a href="#">In den Beispielen für Abfragemethoden.</a></li> <li>• <a href="#">Im Lombok-Beispiel.</a></li> <li>• <a href="#">Bei unveränderlichen Klassen.</a></li> </ul>
DynamoDbSortKey	Attribut	Markiert ein Attribut als optionalen Primärsortierschlüssel (Bereichsschlüssel).	<ul style="list-style-type: none"> <li>• <a href="#">Abschnitt „Erste Schritte“ zur Kundenklasse.</a></li> <li>• <a href="#">Bei unveränderlichen Klassen.</a></li> <li>• <a href="#">Im Lombok-Beispiel.</a></li> <li>• <a href="#">In den Beispielen für Abfragemethoden.</a></li> </ul>

Name der Anmerkung	Anmerkung gilt für*	Was es macht	Wo es in diesem Handbuch gezeigt wird
DynamoDbUpdateBehavior	Attribut	Gibt das Verhalten an, wenn dieses Attribut im Rahmen einer „Aktualisierungs“-Operation wie aktualisiert wird UpdateItem.	<a href="#">Einführung und Beispiel.</a>
DynamoDbVersionAttribute	Attribut	Inkrementiert eine Artikelversionsnummer.	<a href="#">Einführung und Diskussion.</a>

\*Sie können Anmerkungen auf Attributebene auf den Getter oder Setter anwenden, aber nicht auf beide. Dieses Handbuch zeigt Anmerkungen zu Gettern.

## Arbeiten mit Amazon EC2

Dieser Abschnitt enthält Beispiele für die Programmierung [Amazon EC2](#), die AWS SDK for Java 2.x verwenden.

### Themen

- [Amazon EC2 Instanzen verwalten](#)
- [Verwenden von AWS-Regionen und Availability Zones](#)
- [Arbeiten mit Sicherheitsgruppen in Amazon EC2](#)
- [Mit Amazon EC2 EC2-Instance-Metadaten arbeiten](#)

# Amazon EC2Instanzen verwalten

## Erstellen einer -Instance

Erstellen Sie eine neue Amazon EC2 Instance, indem Sie die [runInstances](#) Methode des [Ec2Client](#) aufrufen und ihr ein zu [RunInstancesRequest](#) verwendendes [Amazon Machine Image \(AMI\)](#) und einen [Instance-Typ](#) zur Verfügung stellen.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.InstanceType;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

### Code

```
public static String createEC2Instance(Ec2Client ec2, String name, String amiId ) {

    RunInstancesRequest runRequest = RunInstancesRequest.builder()
        .imageId(amiId)
        .instanceType(InstanceType.T1_MICRO)
        .maxCount(1)
        .minCount(1)
        .build();

    RunInstancesResponse response = ec2.runInstances(runRequest);
    String instanceId = response.instances().get(0).instanceId();

    Tag tag = Tag.builder()
        .key("Name")
        .value(name)
        .build();

    CreateTagsRequest tagRequest = CreateTagsRequest.builder()
        .resources(instanceId)
        .tags(tag)
        .build();
```



```
try {
    ec2.createTags(tagRequest);
    System.out.printf(
        "Successfully started EC2 Instance %s based on AMI %s",
        instanceId, amiId);

    return instanceId;

} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

return "";
```

Das [vollständige Beispiel](#) finden Sie unter. GitHub

## Starten einer Instance

Um eine Amazon EC2 Instanz zu starten, rufen Sie die [startInstances](#) Methode des Ec2Client auf und geben ihr eine, die die ID der zu startenden Instanz [StartInstancesRequest](#) enthält.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
```

### Code

```
public static void startInstance(Ec2Client ec2, String instanceId) {

    StartInstancesRequest request = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    ec2.startInstances(request);
    System.out.printf("Successfully started instance %s", instanceId);
}
```

Das [vollständige Beispiel](#) finden Sie unter. GitHub

## Anhalten einer Instance

Um eine Amazon EC2 Instanz zu stoppen, rufen Sie die [stopInstances](#) Methode des Ec2Client auf und geben ihr eine, die die ID der Instanz [StopInstancesRequest](#) enthält, die gestoppt werden soll.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
```

### Code

```
public static void stopInstance(Ec2Client ec2, String instanceId) {

    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    ec2.stopInstances(request);
    System.out.printf("Successfully stopped instance %s", instanceId);
}
```

Das [vollständige Beispiel](#) finden Sie unter. [GitHub](#)

## Neustarten einer Instance

Um eine Amazon EC2 Instanz neu zu starten, rufen Sie die [rebootInstances](#) Methode des Ec2Client auf und geben Sie ihr eine, die die ID der Instanz [RebootInstancesRequest](#) enthält, die neu gestartet werden soll.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.RebootInstancesRequest;
```

### Code

```
public static void rebootEC2Instance(Ec2Client ec2, String instanceId) {
```

```
try {
    RebootInstancesRequest request = RebootInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    ec2.rebootInstances(request);
    System.out.printf(
        "Successfully rebooted instance %s", instanceId);
} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Das [vollständige Beispiel](#) finden Sie unter. [GitHub](#)

## Beschreiben von Instances

Um Ihre Instanzen aufzulisten, erstellen Sie eine Methode des `Ec2Client`

[DescribeInstancesRequest](#) und rufen Sie sie auf. [describeInstances](#) Es wird ein [DescribeInstancesResponse](#) Objekt zurückgegeben, mit dem Sie die Amazon EC2 Instanzen für Ihr Konto und Ihre Region auflisten können.

Instances werden nach Reservierung gruppiert. Jede Reservierung entspricht dem Aufruf von `startInstances`, durch den die Instance gestartet wurde. Um Ihre Instances aufzulisten, müssen Sie zuerst die Methode `reservations` der Klasse `DescribeInstancesResponse` aufrufen und dann `instances` für jedes zurückgegebene [Reservation](#)-Objekt aufrufen.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Instance;
import software.amazon.awssdk.services.ec2.model.Reservation;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

### Code

```
public static void describeEC2Instances( Ec2Client ec2){
```

```
String nextToken = null;

try {
    do {
        DescribeInstancesRequest request =
DescribeInstancesRequest.builder().maxResults(6).nextToken(nextToken).build();
        DescribeInstancesResponse response = ec2.describeInstances(request);

        for (Reservation reservation : response.reservations()) {
            for (Instance instance : reservation.instances()) {
                System.out.println("Instance Id is " + instance.instanceId());
                System.out.println("Image id is "+ instance.imageId());
                System.out.println("Instance type is "+
instance.instanceType());
                System.out.println("Instance state name is "+
instance.state().name());
                System.out.println("monitoring information is "+
instance.monitoring().state());

            }
        }
        nextToken = response.nextToken();
    } while (nextToken != null);

} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Die Ergebnisse werden seitenweise zurückgegeben. Sie können die weiteren Ergebnisse abrufen, indem Sie den von der `nextToken`-Methode des Rückgabeobjekts zurückgegebenen Wert an die `nextToken`-Methode eines neuen Anforderungsobjekts übergeben. Verwenden Sie dann das neue Anforderungsobjekt für den nächsten Aufruf von `describeInstances`.

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Überwachen einer Instance

Sie können verschiedene Aspekte Ihrer Amazon EC2-Instances überwachen, wie CPU- und Netzwerkauslastung, verfügbarer Arbeitsspeicher und verbleibender Festplattenspeicher. Weitere

Informationen zur Instanzüberwachung finden Sie unter [Überwachung Amazon EC2](#) im Amazon EC2 Benutzerhandbuch für Linux-Instances.

Um mit der Überwachung einer Instanz zu beginnen, müssen Sie eine [MonitorInstancesRequest](#) mit der ID der zu überwachenden Instanz erstellen und sie an die Methode des Ec2Client übergeben.

[monitorInstances](#)

## Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.MonitorInstancesRequest;
import software.amazon.awssdk.services.ec2.model.UnmonitorInstancesRequest;
```

## Code

```
public static void monitorInstance( Ec2Client ec2, String instanceId) {

    MonitorInstancesRequest request = MonitorInstancesRequest.builder()
        .instanceIds(instanceId).build();

    ec2.monitorInstances(request);
    System.out.printf(
        "Successfully enabled monitoring for instance %s",
        instanceId);
}
```

Das [vollständige Beispiel](#) finden Sie unter. GitHub

## Anhalten der Instance-Überwachung

Um die Überwachung einer Instanz zu beenden, erstellen Sie eine [UnmonitorInstancesRequest](#) mit der ID der Instanz, deren Überwachung beendet werden soll, und übergeben Sie sie an die Methode des [unmonitorInstances](#) Ec2Client.

## Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.MonitorInstancesRequest;
```

```
import software.amazon.awssdk.services.ec2.model.UnmonitorInstancesRequest;
```

## Code

```
public static void unmonitorInstance(Ec2Client ec2, String instanceId) {
    UnmonitorInstancesRequest request = UnmonitorInstancesRequest.builder()
        .instanceIds(instanceId).build();

    ec2.unmonitorInstances(request);

    System.out.printf(
        "Successfully disabled monitoring for instance %s",
        instanceId);
}
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Weitere Informationen

- [RunInstances](#) in der Amazon EC2 API-Referenz
- [DescribeInstances](#) in der Amazon EC2 API-Referenz
- [StartInstances](#) in der Amazon EC2 API-Referenz
- [StopInstances](#) in der Amazon EC2 API-Referenz
- [RebootInstances](#) in der Amazon EC2 API-Referenz
- [MonitorInstances](#) in der Amazon EC2 API-Referenz
- [UnmonitorInstances](#) in der Amazon EC2 API-Referenz

## Verwenden von AWS-Regionen und Availability Zones

### Beschreiben von Regionen

Um die für Ihr Konto verfügbaren Regionen aufzulisten, rufen Sie die Methode des `Ec2Client` auf `describeRegions`. Sie gibt [DescribeRegionsResponse](#) zurück. Rufen Sie die `regions`-Methode des zurückgegebenen Objekts auf und Sie erhalten eine Liste mit [Region](#)-Objekten, von denen jedes für eine Region steht.

### Importe

```
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

## Code

```
try {
    DescribeRegionsResponse regionsResponse = ec2.describeRegions();
    regionsResponse.regions().forEach(region -> {
        System.out.printf(
            "Found Region %s with endpoint %s%n",
            region.regionName(),
            region.endpoint());
        System.out.println();
    });
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Beschreiben von Availability Zones

Um jede Availability Zone aufzulisten, die Ihrem Konto zur Verfügung steht, rufen Sie die Methode des Ec2Client auf `describeAvailabilityZones`. Sie gibt [DescribeAvailabilityZonesResponse](#) zurück. Rufen Sie die `availabilityZones` Methode auf, um eine Liste der [AvailabilityZone](#) Objekte abzurufen, die jede Availability Zone repräsentieren.

## Importe

```
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

## Code

Erstellen Sie den Ec2Client .

```
software.amazon.awssdk.regions.Region region =
software.amazon.awssdk.regions.Region.US_EAST_1;
Ec2Client ec2 = Ec2Client.builder()
    .region(region)
    .build();
```

Rufen Sie dann `describeAvailabilityZones()` auf und rufen Sie Ergebnisse ab.

```
DescribeAvailabilityZonesResponse zonesResponse =
ec2.describeAvailabilityZones();
zonesResponse.availabilityZones().forEach(zone -> {
    System.out.printf(
        "Found Availability Zone %s with status %s in region %s\n",
        zone.zoneName(),
        zone.state(),
        zone.regionName()
    );
    System.out.println();
});
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Beschreiben von Konten

Um EC2-related Informationen zu Ihrem Konto aufzulisten, rufen Sie die Methode des `Ec2Client` auf `describeAccountAttributes`. Diese Methode gibt ein `-DescribeAccountAttributesResponse` Objekt zurück. Rufen Sie diese `accountAttributes` Objektmethode auf, um eine Liste von `AccountAttribute` Objekten abzurufen. Sie können die Liste durchlaufen, um ein `AccountAttribute` Objekt abzurufen.

Sie können die Attributwerte Ihres Kontos abrufen, indem Sie die `attributeValues` Methode des `AccountAttribute` Objekts aufrufen. Diese Methode gibt eine Liste von `AccountAttributeValue` Objekten zurück. Sie können diese zweite Liste durchlaufen, um den Wert von Attributen anzuzeigen (siehe das folgende Codebeispiel).

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```



## Code

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAccount {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .build();

        describeEC2Account(ec2);
        System.out.print("Done");
        ec2.close();
    }

    public static void describeEC2Account(Ec2Client ec2) {
        try {
            DescribeAccountAttributesResponse accountResults =
ec2.describeAccountAttributes();
            accountResults.accountAttributes().forEach(attribute -> {
                System.out.print("\n The name of the attribute is " +
attribute.attributeName());
                attribute.attributeValues().forEach(
                    myValue -> System.out.print("\n The value of the attribute is "
+ myValue.attributeValue()));
            });

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Weitere Informationen

- [Regionen und Availability Zones](#) im Amazon EC2 Benutzerhandbuch für Linux-Instances
- [DescribeRegions](#) in der Amazon EC2 API-Referenz zu
- [DescribeAvailabilityZones](#) in der Amazon EC2 API-Referenz zu

## Arbeiten mit Sicherheitsgruppen in Amazon EC2

### Eine Sicherheitsgruppe erstellen

Um eine Sicherheitsgruppe zu erstellen, rufen Sie die `createSecurityGroup`-Methode des `Ec2Client` mit einem [auf `CreateSecurityGroupRequest`](#), der den Namen des Schlüssels enthält. `createSecurityGroup`

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.IpRange;
```

### Code

```
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
    .groupName(groupName)
    .description(groupDesc)
    .vpcId(vpcId)
    .build();

        CreateSecurityGroupResponse resp= ec2.createSecurityGroup(createRequest);
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Konfigurieren einer Sicherheitsgruppe

Eine Sicherheitsgruppe kann sowohl eingehenden (eingehenden) als auch ausgehenden (ausgehenden) Datenverkehr zu Ihren Amazon EC2 Instances steuern.

Um Ihrer Sicherheitsgruppe Regeln für eingehenden Datenverkehr hinzuzufügen, verwenden Sie die `authorizeSecurityGroupIngress`-Methode des `Ec2Client` und geben Sie den Namen der Sicherheitsgruppe und die Zugriffsregeln (`IpPermission`) an, die Sie ihr innerhalb eines `AuthorizeSecurityGroupIngressRequest` Objekts zuweisen möchten. `authorizeSecurityGroupIngress` Im folgenden Beispiel wird gezeigt, wie Sie einer Sicherheitsgruppe IP-Berechtigungen hinzufügen.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.IpRange;
```

### Code

Erstellen Sie zunächst einen `Ec2Client`

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

Verwenden Sie dann die `authorizeSecurityGroupIngress`-Methode von `Ec2Client`,

```
IpRange ipRange = IpRange.builder()
    .cidrIp("0.0.0.0/0").build();

IpPermission ipPerm = IpPermission.builder()
    .ipProtocol("tcp")
    .toPort(80)
```

```
        .fromPort(80)
        .ipRanges(ipRange)
        .build();

    IpPermission ipPerm2 = IpPermission.builder()
        .ipProtocol("tcp")
        .toPort(22)
        .fromPort(22)
        .ipRanges(ipRange)
        .build();

    AuthorizeSecurityGroupIngressRequest authRequest =
        AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(groupName)
            .ipPermissions(ipPerm, ipPerm2)
            .build();

    AuthorizeSecurityGroupIngressResponse authResponse =
        ec2.authorizeSecurityGroupIngress(authRequest);

    System.out.printf(
        "Successfully added ingress policy to Security Group %s",
        groupName);

    return resp.groupId();

} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
```

Um der Sicherheitsgruppe eine Ausgangsregel hinzuzufügen, geben Sie ähnliche Daten in einem [AuthorizeSecurityGroupEgressRequest](#) wie in der `authorizeSecurityGroupEgress` Methode des `Ec2Client` an.

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Beschreiben von Sicherheitsgruppen

Um Ihre Sicherheitsgruppen zu beschreiben oder Informationen darüber zu erhalten, rufen Sie die Methode des `Ec2Client` `describeSecurityGroups`. Es gibt einen

zurück [DescribeSecurityGroupsResponse](#), mit dem Sie auf die Liste der Sicherheitsgruppen zugreifen können, indem Sie die Methode `describeSecurityGroups`, die eine Liste von [SecurityGroup](#) Objekten zurückgibt.

## Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.SecurityGroup;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

## Code

```
public static void describeEC2SecurityGroups(Ec2Client ec2, String groupId) {

    try {
        DescribeSecurityGroupsRequest request =
            DescribeSecurityGroupsRequest.builder()
                .groupIds(groupId).build();

        DescribeSecurityGroupsResponse response =
            ec2.describeSecurityGroups(request);

        for(SecurityGroup group : response.securityGroups()) {
            System.out.printf(
                "Found Security Group with id %s, " +
                "vpc id %s " +
                "and description %s",
                group.groupId(),
                group.vpcId(),
                group.description());
        }
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Löschen einer Sicherheitsgruppe

Um eine Sicherheitsgruppe zu löschen, rufen Sie die `deleteSecurityGroup`-Methode des `Ec2Client` auf und übergeben Sie ihr eine [DeleteSecurityGroupRequest](#), die die ID der zu löschenden Sicherheitsgruppe enthält.

`deleteSecurityGroup`

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

### Code

```
public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {

    try {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();

        ec2.deleteSecurityGroup(request);
        System.out.printf(
            "Successfully deleted Security Group with id %s", groupId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

### Weitere Informationen

- [Amazon EC2 Sicherheitsgruppen](#) im Amazon EC2 Benutzerhandbuch für Linux-Instances
- [Autorisieren von eingehendem Datenverkehr für Ihre Linux-Instances](#) im Amazon EC2 Benutzerhandbuch für Linux-Instances
- [CreateSecurityGroup](#) in der Amazon EC2 API-Referenz zu

- [DescribeSecurityGroups](#) in der Amazon EC2 API-Referenz zu
- [DeleteSecurityGroup](#) in der Amazon EC2 API-Referenz zu
- [AuthorizeSecurityGroupIngress](#) in der Amazon EC2 API-Referenz zu

## Mit Amazon EC2 EC2-Instance-Metadaten arbeiten

Ein Java SDK-Client für den Amazon EC2 Instance Metadata Service (Metadaten-Client) ermöglicht Ihren Anwendungen den Zugriff auf Metadaten auf ihrer lokalen EC2-Instance. Der Metadaten-Client arbeitet mit der lokalen Instanz von [IMDSv2](#) (Instance Metadata Service v2) und verwendet sitzungorientierte Anfragen.

Zwei Client-Klassen sind im SDK verfügbar. Die synchrone [Ec2MetadataClient](#) ist für blockierende Operationen und die [Ec2MetadataAsyncClient](#) für asynchrone, nicht blockierende Anwendungsfälle vorgesehen.

### Erste Schritte

Um den Metadaten-Client zu verwenden, fügen Sie das `imds` Maven-Artefakt zu Ihrem Projekt hinzu. Sie benötigen auch Klassen für eine [SdkHttpClient](#) (oder eine [SdkAsyncHttpClient](#) für die asynchrone Variante) im Klassenpfad.

Das folgende Maven-XML zeigt Abhängigkeitsschnipsel für die Verwendung von Synchron [URLConnectionHttpClient](#) zusammen mit der Abhängigkeit für Metadaten-Clients.

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

<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>imds</artifactId>
```

```

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

```

Suchen Sie im [zentralen Maven-Repository](#) nach der neuesten Version desbom Artefakts.

Um einen asynchronen HTTP-Client zu verwenden, ersetzen Sie das Abhängigkeits-Snippet für das `url-connection-client` Artefakt. Das folgende Snippet enthält beispielsweise die [NettyNioAsyncHttpClient](#) Implementierung.

```

<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>netty-nio-client</artifactId>
</dependency>

```

## Verwenden Sie den Metadaten-Client

Instanzieren Sie einen Metadaten-Client

Sie können eine Instanz eines Synchronous instanziiieren, `Ec2MetadataClient` wenn nur eine Implementierung der `SdkHttpClient` Schnittstelle im Klassenpfad vorhanden ist. Rufen Sie dafür die statische `Ec2MetadataClient#create()` Methode auf, wie im folgenden Snippet gezeigt.

```

Ec2MetadataClient client = Ec2MetadataClient.create(); //
'Ec2MetadataAsyncClient#create' is the asynchronous version.

```

Wenn Ihre Anwendung mehrere Implementierungen der `SdkHttpClient` `SdkHttpAsyncClient` OR-Schnittstelle hat, müssen Sie eine Implementierung angeben, die der Metadaten-Client verwenden soll, wie im [the section called "Konfigurierbarer HTTP-Client"](#) Abschnitt gezeigt.

### Note

Für die meisten Service-Clients, wie Amazon S3, fügt das SDK for Java automatisch Implementierungen der `SdkHttpClient` `SdkHttpAsyncClient` OR-Schnittstelle hinzu. Wenn Ihr Metadaten-Client dieselbe Implementierung verwendet, `Ec2MetadataClient#create()` funktioniert das. Wenn Sie eine andere



Implementierung benötigen, müssen Sie diese angeben, wenn Sie den Metadaten-Client erstellen.

## Anfragen senden

Um Instanzmetadaten abzurufen, instanzieren Sie die `EC2MetadataClient` Klasse und rufen Sie die `get` Methode mit einem Pfadparameter auf, der die [Metadatenkategorie der Instanz](#) angibt.

Das folgende Beispiel druckt den Wert, der dem `ami-id` Schlüssel zugeordnet ist, an die Konsole.

```
Ec2MetadataClient client = Ec2MetadataClient.create();
Ec2MetadataResponse response = client.get("/latest/meta-data/ami-id");
System.out.println(response.asString());
client.close(); // Closes the internal resources used by the Ec2MetadataClient class.
```

Wenn der Pfad nicht gültig ist, löst die `get` Methode eine Ausnahme aus.

Verwenden Sie dieselbe Client-Instanz für mehrere Anfragen wieder, rufen Sie den Client jedoch `close` an, wenn er nicht mehr benötigt wird, um Ressourcen freizugeben. Nach dem Aufruf der `close`-Methode kann die Client-Instanz nicht mehr verwendet werden.

## Antworten analysieren

EC2-Instanzmetadaten können in verschiedenen Formaten ausgegeben werden. Klartext und JSON sind die am häufigsten verwendeten Formate. Die Metadaten-Clients bieten Möglichkeiten, mit diesen Formaten zu arbeiten.

Wie das folgende Beispiel zeigt, verwenden Sie die `asString` Methode, um die Daten als Java-Zeichenfolge abzurufen. Sie können die `asList` Methode auch verwenden, um eine Klartextantwort zu trennen, die mehrere Zeilen zurückgibt.

```
Ec2MetadataClient client = Ec2MetadataClient.create();
Ec2MetadataResponse response = client.get("/latest/meta-data/");
String fullResponse = response.asString();
List<String> splits = response.asList();
```

Wenn die Antwort in JSON ist, verwenden Sie die `Ec2MetadataResponse#asDocument` Methode, um die JSON-Antwort in eine [Dokumentinstanz](#) zu analysieren, wie im folgenden Codeausschnitt gezeigt.

```
Document fullResponse = response.asDocument();
```

Eine Ausnahme wird ausgelöst, wenn das Format der Metadaten nicht in JSON ist. Wenn die Antwort erfolgreich analysiert wurde, können Sie die [Dokument-API](#) verwenden, um die Antwort genauer zu überprüfen. Sehen Sie sich das [Kategoriediagramm für Instanzmetadaten](#) an, um zu erfahren, welche Metadatenkategorien Antworten im JSON-Format liefern.

## Einen Metadaten-Client konfigurieren

### Wiederholversuche

Sie können einen Metadaten-Client mit einem Wiederholungsmechanismus konfigurieren. Wenn Sie dies tun, kann der Client Anfragen, die aus unerwarteten Gründen fehlschlagen, automatisch wiederholen. Standardmäßig versucht es der Client bei einer fehlgeschlagenen Anfrage dreimal mit einer exponentiellen Backoff-Zeit zwischen den Versuchen.

Wenn Ihr Anwendungsfall einen anderen Wiederholungsmechanismus erfordert, können Sie den Client mithilfe der `retryPolicy` Methode in seinem Builder anpassen. Das folgende Beispiel zeigt beispielsweise einen synchronen Client, der mit einer festen Verzögerung von zwei Sekunden zwischen Versuchen und fünf Wiederholungsversuchen konfiguriert ist.

```
BackoffStrategy fixedBackoffStrategy =
    FixedDelayBackoffStrategy.create(Duration.ofSeconds(2));
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .retryPolicy(retryPolicyBuilder ->
            retryPolicyBuilder.numRetries(5)

            .backoffStrategy(fixedBackoffStrategy))
        .build();
```

Es gibt mehrere [BackoffStrategien](#), die Sie mit einem Metadaten-Client verwenden können.

Sie können den Wiederholungsmechanismus auch vollständig deaktivieren, wie das folgende Snippet zeigt.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .retryPolicy(Ec2MetadataRetryPolicy.none())
        .build();
```

Bei Verwendung `Ec2MetadataRetryPolicy#none()` wird die standardmäßige Wiederholungsrichtlinie deaktiviert, sodass der Metadaten-Client keine Wiederholungsversuche unternimmt.

## IP-Version

Standardmäßig verwendet ein Metadaten-Client den IPV4-Endpunkt unter `http://169.254.169.254`. Um den Client so zu ändern, dass er die IPV6-Version verwendet, verwenden Sie entweder die `endpointMode` oder die `endpoint` Methode des Builders. Eine Ausnahme entsteht, wenn beide Methoden im Builder aufgerufen werden.

Die folgenden Beispiele zeigen beide IPv6-Optionen.

```
Ec2MetadataClient client =  
    Ec2MetadataClient.builder()  
        .endpointMode(EndpointMode.IPV6)  
        .build();
```

```
Ec2MetadataClient client =  
    Ec2MetadataClient.builder()  
        .endpoint(URI.create("http://[fd00:ec2::254]"))  
        .build();
```

## Schlüsselfunktionen

### Asynchroner Client

Um die nicht blockierende Version des Clients zu verwenden, instanziiieren Sie eine Instanz der `Ec2MetadataAsyncClient` Klasse. Der Code im folgenden Beispiel erstellt einen asynchronen Client mit Standardeinstellungen und verwendet die `get` Methode, um den Wert für den `ami-id` Schlüssel abzurufen.

```
Ec2MetadataAsyncClient asyncClient = Ec2MetadataAsyncClient.create();  
CompletableFuture<Ec2MetadataResponse> response = asyncClient.get("/latest/meta-data/  
ami-id");
```

Das von der `get` Methode `java.util.concurrent.CompletableFuture` zurückgegebene Ergebnis ist abgeschlossen, wenn die Antwort zurückgegeben wird. Im folgenden Beispiel werden die `ami-id` Metadaten auf die Konsole gedruckt.

```
response.thenAccept(metadata -> System.out.println(metadata.asString()));
```

## Konfigurierbarer HTTP-Client

Der Builder für jeden Metadaten-Client verfügt über eine `httpClient` Methode, mit der Sie einen benutzerdefinierten HTTP-Client bereitstellen können.

Das folgende Beispiel zeigt den Code für eine benutzerdefinierte `URLConnectionHttpClient` Instanz.

```
SdkHttpClient httpClient =
    UrlConnectionHttpClient.builder()
        .socketTimeout(Duration.ofMinutes(5))
        .proxyConfiguration(proxy ->
            proxy.endpoint(URI.create("http://proxy.example.net:8888")))
        .build();
Ec2MetadataClient metaDataClient =
    Ec2MetadataClient.builder()
        .httpClient(httpClient)
        .build();
// Use the metaDataClient instance.
metaDataClient.close(); // Close the instance when no longer needed.
```

Das folgende Beispiel zeigt Code für eine benutzerdefinierte `NettyNioAsyncHttpClient` Instanz mit einem asynchronen Metadatenclient.

```
SdkAsyncHttpClient httpAsyncClient =
    NettyNioAsyncHttpClient.builder()
        .connectionTimeout(Duration.ofMinutes(5))
        .maxConcurrency(100)
        .build();
Ec2MetadataAsyncClient asyncMetaDataClient =
    Ec2MetadataAsyncClient.builder()
        .httpClient(httpAsyncClient)
        .build();
// Use the asyncMetaDataClient instance.
asyncMetaDataClient.close(); // Close the instance when no longer needed.
```

Das [the section called "HTTP-Clients"](#) Thema in diesem Handbuch enthält Einzelheiten zur Konfiguration der HTTP-Clients, die im SDK for Java verfügbar sind.

## Zwischenspeichern von Token

Da die Metadaten-Clients IMDSv2 verwenden, sind alle Anfragen einer Sitzung zugeordnet. Eine Sitzung wird durch ein Token mit einem Ablaufdatum definiert, das der Metadaten-Client für Sie verwaltet. Jede Metadatenanfrage verwendet das Token automatisch wieder, bis es abläuft.

Standardmäßig dauert ein Token sechs Stunden (21 600 Sekunden). Wir empfehlen, den time-to-live Standardwert, es sei denn, Ihr spezieller Anwendungsfall erfordert, eine erweiterte Konfiguration.

Konfigurieren Sie die Dauer bei Bedarf mithilfe der `tokenTtl` Builder-Methode. Der Code im folgenden Codeausschnitt erstellt beispielsweise einen Client mit einer Sitzungsdauer von fünf Minuten.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .tokenTtl(Duration.ofMinutes(5))
        .build();
```

Wenn Sie den Aufruf der `tokenTtl` Methode im Builder auslassen, wird stattdessen die Standarddauer von 21.600 verwendet.

## Arbeiten mit IAM

Dieser Abschnitt enthält Beispiele für die Programmierung AWS Identity and Access Management (IAM) mit AWS SDK for Java 2.x.

AWS Identity and Access Management (IAM) ermöglicht es Ihnen, den Zugriff Ihrer Benutzer auf AWS Dienste und Ressourcen sicher zu kontrollieren. Mithilfe von IAM Cookies können Sie AWS Benutzer und Gruppen erstellen und verwalten und ihnen mithilfe von Berechtigungen den Zugriff auf AWS Ressourcen gewähren oder verweigern. Eine vollständige Anleitung dazu IAM finden Sie im [IAM Benutzerhandbuch](#).

Die folgenden Beispiele enthalten nur den Code, der zur Demonstration jeder Technik nötig ist. Der [vollständige Beispielcode ist verfügbar unter GitHub](#). Von dort aus können Sie eine einzelne Quelldatei herunterladen oder das Repository klonen, um alle Beispiele lokal zu erstellen und auszuführen.

### Themen

- [Verwalten von IAM Zugriffsschlüsseln](#)
- [IAM Benutzer verwalten](#)

- [Erstellen von IAM-Richtlinien mit der AWS SDK for Java 2.x](#)
- [Arbeiten mit IAM Richtlinien](#)
- [Arbeiten mit IAM Serverzertifikaten](#)

## Verwalten von IAM Zugriffsschlüsseln

### Erstellen eines Zugriffsschlüssels

Um einen IAM Zugriffsschlüssel zu erstellen, rufen Sie die `-IamClient`'s `createAccessKey` Methode mit einem `-CreateAccessKeyRequest` Objekt auf.

#### Note

Sie müssen die Region auf `AWS_GLOBAL` setzen, damit `IamClient` Aufrufe funktionieren, da ein globaler Service IAM ist.

### Importe

```
import software.amazon.awssdk.services.iam.model.CreateAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

### Code

```
public static String createIAMAccessKey(IamClient iam, String user) {

    try {
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
            .userName(user).build();

        CreateAccessKeyResponse response = iam.createAccessKey(request);
        String keyId = response.accessKey().accessKeyId();
        return keyId;

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
    return "";  
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Auflisten von Zugriffsschlüsseln

Um die Zugriffsschlüssel für einen bestimmten Benutzer aufzulisten, erstellen Sie ein [ListAccessKeysRequest](#) Objekt, das den Benutzernamen enthält, für den Schlüssel aufgelistet werden sollen, und übergeben Sie es an die `IamClient`'s `listAccessKeys` Methode .

### Note

Wenn Sie keinen Benutzernamen für angeben `listAccessKeys`, wird versucht, die Zugriffsschlüssel aufzulisten, die dem zugeordnet sind AWS-Konto , der die Anforderung signiert hat.

## Importe

```
import software.amazon.awssdk.services.iam.model.AccessKeyMetadata;  
import software.amazon.awssdk.services.iam.model.IamException;  
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;  
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.iam.IamClient;
```

## Code

```
public static void listKeys( IamClient iam,String userName ){  
  
    try {  
        boolean done = false;  
        String newMarker = null;  
  
        while (!done) {  
            ListAccessKeysResponse response;  
  
            if(newMarker == null) {  
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
```

```
        .userName(userName).build();
        response = iam.listAccessKeys(request);
    } else {
        ListAccessKeysRequest request = ListAccessKeysRequest.builder()
            .userName(userName)
            .marker(newMarker).build();
        response = iam.listAccessKeys(request);
    }

    for (AccessKeyMetadata metadata :
        response.accessKeyMetadata()) {
        System.out.format("Retrieved access key %s",
            metadata.accessKeyId());
    }

    if (!response.isTruncated()) {
        done = true;
    } else {
        newMarker = response.marker();
    }
}

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Die Ergebnisse von `listAccessKeys` sind seitenweise angeordnet (mit einem Standardhöchstwert von 100 Datensätzen pro Aufruf). Sie können `isTruncated` für das zurückgegebene [ListAccessKeysResponse](#) Objekt aufrufen, um festzustellen, ob die Abfrage weniger Ergebnisse zurückgegeben hat und verfügbar ist. Falls ja, rufen Sie `marker` für `ListAccessKeysResponse` auf und verwenden es beim Erstellen einer neuen Anforderung. Verwenden Sie diese neue Anforderung im nächsten Aufruf von `listAccessKeys`.

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Abrufen der letzten Nutzungszeit eines Zugriffsschlüssels

Um die Zeit zu erhalten, zu der ein Zugriffsschlüssel zuletzt verwendet wurde, rufen Sie die `IamClient`'s `getAccessKeyLastUsed` Methode mit der ID des Zugriffsschlüssels auf (die mit einem [GetAccessKeyLastUsedRequest](#) Objekt übergeben werden kann).



Anschließend können Sie das zurückgegebene [GetAccessKeyLastUsedResponse](#) Objekt verwenden, um die letzte Nutzungszeit des Schlüssels abzurufen.

## Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetAccessKeyLastUsedRequest;
import software.amazon.awssdk.services.iam.model.GetAccessKeyLastUsedResponse;
import software.amazon.awssdk.services.iam.model.IamException;
```

## Code

```
public static void getAccessKeyLastUsed(IamClient iam, String accessId ){

    try {
        GetAccessKeyLastUsedRequest request = GetAccessKeyLastUsedRequest.builder()
            .accessKeyId(accessId).build();

        GetAccessKeyLastUsedResponse response = iam.getAccessKeyLastUsed(request);

        System.out.println("Access key was last used at: " +
            response.accessKeyLastUsed().lastUsedDate());

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Aktivieren oder Deaktivieren von Zugriffsschlüsseln

Sie können einen Zugriffsschlüssel aktivieren oder deaktivieren, indem Sie ein [UpdateAccessKeyRequest](#) Objekt erstellen, die Zugriffsschlüssel-ID, optional den Benutzernamen und die gewünschte angeben und dann das [status](#) Anforderungsobjekt an die `IamClient`'s `updateAccessKey` Methode übergeben.

## Importe

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.StatusType;
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

## Code

```
public static void updateKey(IamClient iam, String username, String accessId,
String status ) {

    try {
        if (status.toLowerCase().equalsIgnoreCase("active")) {
            statusType = StatusType.ACTIVE;
        } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
            statusType = StatusType.INACTIVE;
        } else {
            statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
        }
        UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
            .accessKeyId(accessId)
            .userName(username)
            .status(statusType)
            .build();

        iam.updateAccessKey(request);

        System.out.printf(
            "Successfully updated the status of access key %s to" +
            "status %s for user %s", accessId, status, username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Löschen eines Zugriffsschlüssels

Um einen Zugriffsschlüssel dauerhaft zu löschen, rufen Sie die `IamClient`'s `deleteKey` Methode auf und stellen Sie ihr eine mit der ID und dem Benutzernamen des Zugriffsschlüssels

[DeleteAccessKeyRequest](#) zur Verfügung.

### Note

Nach dem Löschen können Schlüssel nicht mehr abgerufen oder verwendet werden. Um einen Schlüssel vorübergehend zu deaktivieren, damit er später wieder aktiviert werden kann, verwenden Sie stattdessen die [updateAccessKey](#) Methode.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.IamException;
```

### Code

```
public static void deleteKey(IamClient iam ,String username, String accessKey ) {

    try {
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
            .accessKeyId(accessKey)
            .userName(username)
            .build();

        iam.deleteAccessKey(request);
        System.out.println("Successfully deleted access key " + accessKey +
            " from user " + username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Weitere Informationen

- [CreateAccessKey](#) in der IAM API-Referenz zu
- [ListAccessKeys](#) in der IAM API-Referenz zu
- [GetAccessKeyLastUsed](#) in der IAM API-Referenz zu
- [UpdateAccessKey](#) in der IAM API-Referenz zu
- [DeleteAccessKey](#) in der IAM API-Referenz zu

## IAM Benutzer verwalten

### Erstellen eines Benutzers

Erstellen Sie einen neuen IAM Benutzer, indem Sie den Benutzernamen für die Methode `IamClient` des `createUser` mit einem [CreateUserRequest](#) Objekt bereitstellen, das den Benutzernamen enthält.

### Importe

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
import software.amazon.awssdk.services.iam.model.GetUserResponse;
```

### Code

```
public static String createIAMUser(IamClient iam, String username ) {

    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();

        CreateUserRequest request = CreateUserRequest.builder()
            .userName(username)
```

```
        .build();

        CreateUserResponse response = iam.createUser(request);

        // Wait until the user is created
        GetUserRequest userRequest = GetUserRequest.builder()
            .userName(response.user().userName())
            .build();

        WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
        waitUntilUserExists.matched().response().ifPresent(System.out::println);
        return response.user().userName();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## -Benutzer auflisten

Um die IAM Benutzer für Ihr Konto aufzulisten, erstellen Sie einen neuen [ListUsersRequest](#) und übergeben Sie ihn an die `iamClient.listUsers`-Methode. Sie können die Liste der Benutzer abrufen, indem Sie `users` für das zurückgegebene [ListUsersResponse](#) Objekt aufrufen.

Die von `listUsers` zurückgegebene Benutzerliste ist segmentiert. Sie können prüfen, ob weitere Ergebnisse bereitliegen, indem Sie die `isTruncated`-Methode des Antwortobjekts aufrufen. Wenn sie `true` zurückgibt, rufen Sie die `marker()`-Methode des Antwortobjekts auf. Verwenden Sie den Marker-Wert, um ein neues Objekt zu erstellen. Anschließend rufen Sie die `listUsers`-Methode erneut für die neue Anforderung auf.

## Importe

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.services.iam.model.User;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.iam.IamClient;
```

## Code

```
public static void listAllUsers(IamClient iam ) {

    try {

        boolean done = false;
        String newMarker = null;

        while(!done) {
            ListUsersResponse response;

            if (newMarker == null) {
                ListUsersRequest request = ListUsersRequest.builder().build();
                response = iam.listUsers(request);
            } else {
                ListUsersRequest request = ListUsersRequest.builder()
                    .marker(newMarker).build();
                response = iam.listUsers(request);
            }

            for(User user : response.users()) {
                System.out.format("\n Retrieved user %s", user.userName());
            }

            if(!response.isTruncated()) {
                done = true;
            } else {
                newMarker = response.marker();
            }
        }
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Aktualisieren eines Benutzers

Um einen Benutzer zu aktualisieren, rufen Sie die `updateUser` Methode des `IamClient` Objekts auf, die ein [UpdateUserRequest](#) Objekt verwendet, mit dem Sie den Namen oder Pfad des Benutzers ändern können.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;
```

### Code

```
public static void updateIAMUser(IamClient iam, String curName, String newName ) {

    try {
        UpdateUserRequest request = UpdateUserRequest.builder()
            .userName(curName)
            .newUserName(newName)
            .build();

        iam.updateUser(request);
        System.out.printf("Successfully updated user to username %s",
            newName);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Einen Benutzer löschen

Um einen Benutzer zu löschen, rufen Sie die `IamClient deleteUser` Anforderung des mit einem [UpdateUserRequest](#) Objekt auf, das mit dem zu löschenden Benutzernamen festgelegt ist.

### Importe

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
import software.amazon.awssdk.services.iam.model.IamException;
```

## Code

```
public static void deleteIAMUser(IamClient iam, String userName) {

    try {
        DeleteUserRequest request = DeleteUserRequest.builder()
            .userName(userName)
            .build();

        iam.deleteUser(request);
        System.out.println("Successfully deleted IAM user " + userName);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Weitere Informationen

- [IAM Benutzer](#) im IAM -Benutzerhandbuch
- [Verwalten von IAM Benutzern](#) im IAM -Benutzerhandbuch
- [CreateUser](#) in der IAM API-Referenz zu
- [ListUsers](#) in der IAM API-Referenz zu
- [UpdateUser](#) in der IAM API-Referenz zu
- [DeleteUser](#) in der IAM API-Referenz zu

## Erstellen von IAM-Richtlinien mit der AWS SDK for Java 2.x

Die [IAM Policy Builder API](#) ist eine Bibliothek, mit der Sie [IAM-Richtlinien](#) in Java erstellen und in AWS Identity and Access Management (IAM) hochladen können.

Anstatt eine IAM-Richtlinie durch manuelles Zusammenstellen einer JSON-Zeichenfolge oder durch Lesen einer Datei zu erstellen, bietet die API einen clientseitigen, objektorientierten Ansatz zum



Generieren der JSON-Zeichenfolge. Wenn Sie eine vorhandene IAM-Richtlinie im JSON-Format lesen, konvertiert die API sie zur Verarbeitung in eine [IamPolicy](#) Instance.

Die IAM Policy Builder API wurde mit Version 2.20.105 des SDK verfügbar. Verwenden Sie daher diese Version oder eine neuere Version in Ihrer Maven-Build-Datei. Die neueste Versionsnummer des SDK ist [auf Maven Central aufgeführt](#).

Der folgende Codeausschnitt zeigt ein Beispiel für einen Abhängigkeitsblock für eine Maven-pom.xml-Datei. Auf diese Weise können Sie die IAM Policy Builder-API in Ihrem Projekt verwenden.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>iam-policy-builder</artifactId>
  <version>2.20.139</version>
</dependency>
```

## Erstellen eines **IamPolicy**

Dieser Abschnitt zeigt mehrere Beispiele für die Erstellung von Richtlinien mithilfe der IAM-Policy-Builder-API.

Beginnen Sie in jedem der folgenden Beispiele mit [IamPolicy.Builder](#) und fügen Sie eine oder mehrere Anweisungen hinzu, indem Sie die `-addStatement`-Methode verwenden. Nach diesem Muster verfügt der [IamStatement.Builder](#) über Methoden, um der Anweisung die Wirkung, Aktionen, Ressourcen und Bedingungen hinzuzufügen.

Beispiel: Erstellen einer zeitbasierten Richtlinie

Im folgenden Beispiel wird eine identitätsbasierte Richtlinie erstellt, die die Amazon-DynamoDB-GetItem-Aktion zwischen zwei Zeitpunkten zulässt.

```
public String timeBasedPolicyExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:GetItem")
            .addResource(IamResource.ALL)
            .addCondition(b1 -> b1
                .operator(IamConditionOperator.DATE_GREATER_THAN)
                .key("aws:CurrentTime"))
```

```

        .value("2020-04-01T00:00:00Z"))
        .addCondition(b1 -> b1
            .operator(IamConditionOperator.DATE_LESS_THAN)
            .key("aws:CurrentTime")
            .value("2020-06-30T23:59:59Z")))
        .build();

    // Use an IamPolicyWriter to write out the JSON string to a more readable
    format.
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true)
        .build());
}

```

## JSON-Ausgabe

Die letzte Anweisung im vorherigen Beispiel gibt die folgende JSON-Zeichenfolge zurück.

Weitere Informationen zu diesem [Beispiel](#) finden Sie im AWS Identity and Access Management - Benutzerhandbuch.

```

{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Action" : "dynamodb:GetItem",
    "Resource" : "*",
    "Condition" : {
      "DateGreaterThan" : {
        "aws:CurrentTime" : "2020-04-01T00:00:00Z"
      },
      "DateLessThan" : {
        "aws:CurrentTime" : "2020-06-30T23:59:59Z"
      }
    }
  }
}

```

## Beispiel: Angeben mehrerer Bedingungen

Das folgende Beispiel zeigt, wie Sie eine identitätsbasierte Richtlinie erstellen können, die den Zugriff auf bestimmte DynamoDB-Attribute ermöglicht. Die Richtlinie enthält zwei Bedingungen.

```

public String multipleConditionsExample() {

```

```

    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:GetItem")
            .addAction("dynamodb:BatchGetItem")
            .addAction("dynamodb:Query")
            .addAction("dynamodb:PutItem")
            .addAction("dynamodb:UpdateItem")
            .addAction("dynamodb>DeleteItem")
            .addAction("dynamodb:BatchWriteItem")
            .addResource("arn:aws:dynamodb:*:*:table/table-name")

        .addConditions(IamConditionOperator.STRING_EQUALS.addPrefix("ForAllValues:"),
            "dynamodb:Attributes",
            List.of("column-name1", "column-name2", "column-
name3"))

            .addCondition(b1 ->
b1.operator(IamConditionOperator.STRING_EQUALS.addSuffix("IfExists"))
                .key("dynamodb>Select")
                .value("SPECIFIC_ATTRIBUTES"))

        .build();

    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}

```

## JSON-Ausgabe

Die letzte Anweisung im vorherigen Beispiel gibt die folgende JSON-Zeichenfolge zurück.

Weitere Informationen zu diesem [Beispiel](#) finden Sie im AWS Identity and Access Management - Benutzerhandbuch.

```

{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Action" : [ "dynamodb:GetItem", "dynamodb:BatchGetItem", "dynamodb:Query",
"dynamodb:PutItem", "dynamodb:UpdateItem", "dynamodb>DeleteItem",
"dynamodb:BatchWriteItem" ],
    "Resource" : "arn:aws:dynamodb:*:*:table/table-name",
    "Condition" : {
      "ForAllValues:StringEquals" : {

```

```

    "dynamodb:Attributes" : [ "column-name1", "column-name2", "column-name3" ]
  },
  "StringEqualsIfExists" : {
    "dynamodb:Select" : "SPECIFIC_ATTRIBUTES"
  }
}
}

```

## Beispiel: Angeben von Prinzipalen

Das folgende Beispiel zeigt, wie Sie eine ressourcenbasierte Richtlinie erstellen, die den Zugriff auf einen Bucket für alle Prinzipale verweigert, mit Ausnahme der in der Bedingung angegebenen.

```

public String specifyPrincipalsExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.DENY)
            .addAction("s3:*")
            .addPrincipal(IamPrincipal.ALL)
            .addResource("arn:aws:s3:::BUCKETNAME/*")
            .addResource("arn:aws:s3:::BUCKETNAME")
            .addCondition(b1 -> b1
                .operator(IamConditionOperator.ARN_NOT_EQUALS)
                .key("aws:PrincipalArn")
                .value("arn:aws:iam::444455556666:user/user-name")))
        .build();
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}

```

## JSON-Ausgabe

Die letzte Anweisung im vorherigen Beispiel gibt die folgende JSON-Zeichenfolge zurück.

Weitere Informationen zu diesem [Beispiel](#) finden Sie im *AWS Identity and Access Management - Benutzerhandbuch*.

```

{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Deny",
    "Principal" : "*",

```

```

    "Action" : "s3:*",
    "Resource" : [ "arn:aws:s3:::BUCKETNAME/*", "arn:aws:s3:::BUCKETNAME" ],
    "Condition" : {
      "ArnNotEquals" : {
        "aws:PrincipalArn" : "arn:aws:iam::444455556666:user/user-name"
      }
    }
  }
}
}

```

### Beispiel: Kontoübergreifenden Zugriff zulassen

Das folgende Beispiel zeigt, wie Sie einem anderen erlauben AWS-Konto, Objekte in Ihren Bucket hochzuladen, während die volle Eigentümerkontrolle der hochgeladenen Objekte beibehalten wird.

```

public String allowCrossAccountAccessExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addPrincipal(IamPrincipalType.AWS, "111122223333")
            .addAction("s3:PutObject")
            .addResource("arn:aws:s3:::DOC-EXAMPLE-BUCKET/*")
            .addCondition(b1 -> b1
                .operator(IamConditionOperator.STRING_EQUALS)
                .key("s3:x-amz-acl")
                .value("bucket-owner-full-control")))
        .build();
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}

```

### JSON-Ausgabe

Die letzte Anweisung im vorherigen Beispiel gibt die folgende JSON-Zeichenfolge zurück.

Weitere Informationen zu diesem [Beispiel](#) finden Sie im Benutzerhandbuch für Amazon Simple Storage Service.

```

{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Principal" : {

```

```

    "AWS" : "111122223333"
  },
  "Action" : "s3:PutObject",
  "Resource" : "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
  "Condition" : {
    "StringEquals" : {
      "s3:x-amz-acl" : "bucket-owner-full-control"
    }
  }
}
}
}
}

```

## Verwenden eines **IamPolicy** mit IAM

Nachdem Sie eine `IamPolicy` Instance erstellt haben, verwenden Sie eine , [IamClient](#) um mit dem IAM-Service zu arbeiten.

Im folgenden Beispiel wird eine Richtlinie erstellt, die es einer [IAM-Identität](#) ermöglicht, Elemente in eine DynamoDB-Tabelle in dem Konto zu schreiben, das mit dem `-accountID` Parameter angegeben ist. Die Richtlinie wird dann als JSON-Zeichenfolge in IAM hochgeladen.

```

public String createAndUploadPolicyExample(IamClient iam, String accountID, String
policyName) {
    // Build the policy.
    IamPolicy policy =
        IamPolicy.builder() // 'version' defaults to "2012-10-17".
            .addStatement(IamStatement.builder()
                .effect(IamEffect.ALLOW)
                .addAction("dynamodb:PutItem")
                .addResource("arn:aws:dynamodb:us-east-1:" + accountID
+ ":table/exampleTableName")
                .build())
            .build();
    // Upload the policy.
    iam.createPolicy(r ->
r.policyName(policyName).policyDocument(policy.toJson()));
    return policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
}

```

Das nächste Beispiel baut auf dem vorherigen Beispiel auf. Der Code lädt die Richtlinie herunter und verwendet sie als Grundlage für eine neue Richtlinie, indem er die Anweisung kopiert und ändert. Die neue Richtlinie wird dann hochgeladen.

```

    public String createNewBasedOnExistingPolicyExample(IamClient iam, String
accountID, String policyName, String newPolicyName) {

        String policyArn = "arn:aws:iam::" + accountID + ":policy/" + policyName;
        GetPolicyResponse getPolicyResponse = iam.getPolicy(r ->
r.policyArn(policyArn));

        String policyVersion = getPolicyResponse.policy().defaultVersionId();
        GetPolicyVersionResponse getPolicyVersionResponse =
            iam.getPolicyVersion(r ->
r.policyArn(policyArn).versionId(policyVersion));

        // Create an IamPolicy instance from the JSON string returned from IAM.
        String decodedPolicy =
URLDecoder.decode(getPolicyVersionResponse.policyVersion().document(),
StandardCharsets.UTF_8);
        IamPolicy policy = IamPolicy.fromJson(decodedPolicy);

        /*
        All IamPolicy components are immutable, so use the copy method that
creates a new instance that
        can be altered in the same method call.

        Add the ability to get an item from DynamoDB as an additional action.
        */
        IamStatement newStatement = policy.statements().get(0).copy(s ->
s.addAction("dynamodb:GetItem"));

        // Create a new statement that replaces the original statement.
        IamPolicy newPolicy = policy.copy(p ->
p.statements(Arrays.asList(newStatement)));

        // Upload the new policy. IAM now has both policies.
        iam.createPolicy(r -> r.policyName(newPolicyName)
            .policyDocument(newPolicy.toJson()));

        return newPolicy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
    }

```

## IamClient

In den vorherigen Beispielen wird ein `IamClient` Argument verwendet, das wie im folgenden Ausschnitt gezeigt erstellt wird.

```
IamClient iam = IamClient.builder().region(Region.AWS_GLOBAL).build();
```

## Richtlinien in JSON

Die Beispiele geben die folgenden JSON-Zeichenfolgen zurück.

First example

```
{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Action" : "dynamodb:PutItem",
    "Resource" : "arn:aws:dynamodb:us-east-1:111122223333:table/exampleTableName"
  }
}
```

Second example

```
{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Action" : [ "dynamodb:PutItem", "dynamodb:GetItem" ],
    "Resource" : "arn:aws:dynamodb:us-east-1:111122223333:table/exampleTableName"
  }
}
```

## Arbeiten mit IAM Richtlinien

### Erstellen einer Richtlinie

Um eine neue Richtlinie [CreatePolicyRequest](#) zu erstellen, geben Sie den Namen der Richtlinie und ein JSON-formatiertes Richtliniendokument in einem für die `IamClient createPolicy` Methode des `IamClient` an.

### Importe

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.IamException;
```



```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
```

## Code

```
public static String createIAMPolicy(IamClient iam, String policyName ) {

    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();

        CreatePolicyRequest request = CreatePolicyRequest.builder()
            .policyName(policyName)
            .policyDocument(PolicyDocument).build();

        CreatePolicyResponse response = iam.createPolicy(request);

        // Wait until the policy is created
        GetPolicyRequest polRequest = GetPolicyRequest.builder()
            .policyArn(response.policy().arn())
            .build();

        WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);
        waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
        return response.policy().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "" ;
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Abrufen einer Richtlinie

Um eine vorhandene Richtlinie abzurufen, rufen Sie die `-getPolicy` Methode `IamClient` des auf und geben Sie den ARN der Richtlinie in einem `-GetPolicyRequest` Objekt an.

## Importe

```
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

## Code

```
public static void getIAMPolicy(IamClient iam, String policyArn) {

    try {
        GetPolicyRequest request = GetPolicyRequest.builder()
            .policyArn(policyArn).build();

        GetPolicyResponse response = iam.getPolicy(request);
        System.out.format("Successfully retrieved policy %s",
            response.policy().policyName());

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Anfügen einer Rollenrichtlinie

Sie können eine Richtlinie an eine [IAM-Rolle](#) `IamClient` anfügen, indem Sie die `attachRolePolicy`-Methode des aufrufen und ihr den Rollennamen und den Richtlinien-ARN in einer zur Verfügung stellen [AttachRolePolicyRequest](#).

## Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;
```

## Code

```
public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn ) {

    try {

        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
        .roleName(roleName)
        .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

        // Ensure that the policy is not attached to this role
        String polArn = "";
        for (AttachedPolicy policy: attachedPolicies) {
            polArn = policy.policyArn();
            if (polArn.compareTo(policyArn)==0) {
                System.out.println(roleName +
                    " policy is already attached to this role.");
                return;
            }
        }

        AttachRolePolicyRequest attachRequest =
            AttachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(policyArn)
                .build();

        iam.attachRolePolicy(attachRequest);

        System.out.println("Successfully attached policy " + policyArn +
            " to role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    System.out.println("Done");
}
```

```
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Auflisten angefügter Rollenrichtlinien

Listen Sie die angefügten Richtlinien für eine Rolle auf, indem Sie die Methode `IamClient` des `aufrufenListAttachedRolePolicies`. Es benötigt ein [ListAttachedRolePoliciesRequest](#) Objekt, das den Rollennamen enthält, um die Richtlinien aufzulisten.

Rufen Sie für das zurückgegebene [ListAttachedRolePoliciesResponse](#) Objekt `getAttachedPolicies` auf, um die Liste der angehängten Richtlinien abzurufen. Die Ergebnisse sind evtl. gekürzt. Wenn die `isTruncated`-Methode des `ListAttachedRolePoliciesResponse`-Objekts `true` zurückgibt, rufen Sie die `marker`-Methode des `ListAttachedRolePoliciesResponse`-Objekts auf. Verwenden Sie den zurückgegebenen Marker, um eine neue Anforderung zu erstellen, und verwenden Sie sie, um `listAttachedRolePolicies` erneut aufzurufen, um die nächste Ergebnisteilmenge zu erhalten.

## Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;
```

## Code

```
public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn ) {

    try {

        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
        .roleName(roleName)
        .build();
```

```
        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

        // Ensure that the policy is not attached to this role
        String polArn = "";
        for (AttachedPolicy policy: attachedPolicies) {
            polArn = policy.policyArn();
            if (polArn.compareTo(policyArn)==0) {
                System.out.println(roleName +
                    " policy is already attached to this role.");
                return;
            }
        }

        AttachRolePolicyRequest attachRequest =
            AttachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(policyArn)
                .build();

        iam.attachRolePolicy(attachRequest);

        System.out.println("Successfully attached policy " + policyArn +
            " to role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    System.out.println("Done");
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Trennen einer Rollenrichtlinie

Um eine Richtlinie von einer Rolle zu trennen, rufen Sie die `-detachRolePolicy` Methode `IamClient` des auf und geben Sie ihr den Rollennamen und den Richtlinien-ARN in einem `DetachRolePolicyRequest`.

### Importe

```
import software.amazon.awssdk.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

## Code

```
public static void detachPolicy(IamClient iam, String roleName, String policyArn )
{
    try {
        DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(policyArn)
            .build();

        iam.detachRolePolicy(request);
        System.out.println("Successfully detached policy " + policyArn +
            " from role " + roleName);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Weitere Informationen

- [Übersicht über IAM Richtlinien](#) im IAM -Benutzerhandbuch.
- [AWS IAM-Richtlinienreferenz](#) im IAM -Benutzerhandbuch.
- [CreatePolicy](#) in der IAM API-Referenz zu
- [GetPolicy](#) in der IAM API-Referenz zu
- [AttachRolePolicy](#) in der IAM API-Referenz zu
- [ListAttachedRolePolicies](#) in der IAM API-Referenz zu
- [DetachRolePolicy](#) in der IAM API-Referenz zu

## Arbeiten mit IAM Serverzertifikaten

Um HTTPS-Verbindungen zu Ihrer Website oder Anwendung auf zu aktivieren AWS, benötigen Sie ein SSL-/TLS-Serverzertifikat . Sie können ein von bereitgestelltes Serverzertifikat AWS Certificate Manager oder eines verwenden, das Sie von einem externen Anbieter erhalten haben.

Wir empfehlen Ihnen, ACM zum Bereitstellen, Verwalten und Bereitstellen Ihrer Serverzertifikate zu verwenden. Mit können ACM Sie ein Zertifikat anfordern, es für Ihre AWS Ressourcen bereitstellen und Zertifikatserneuerungen für Sie ACM übernehmen lassen. Von bereitgestellte Zertifikate ACM sind kostenlos. Weitere Informationen zu ACM finden Sie im [AWS Certificate Manager - Benutzerhandbuch](#).

### Abrufen eines Serverzertifikats

Sie können ein Serverzertifikat abrufen, indem Sie die Methode `IamClient`des aufrufen `getServerCertificate` und ein [GetServerCertificateRequest](#) mit dem Namen des Zertifikats übergeben.

#### Importe

```
import software.amazon.awssdk.services.iam.model.GetServerCertificateRequest;
import software.amazon.awssdk.services.iam.model.GetServerCertificateResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

#### Code

```
public static void getCertificate(IamClient iam,String certName ) {

    try {
        GetServerCertificateRequest request = GetServerCertificateRequest.builder()
            .serverCertificateName(certName)
            .build();

        GetServerCertificateResponse response = iam.getServerCertificate(request);
        System.out.format("Successfully retrieved certificate with body %s",
            response.serverCertificate().certificateBody());

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Auflisten von Serverzertifikaten

Um Ihre Serverzertifikate aufzulisten, rufen Sie die `-listServerCertificates` Methode `IamClient` des mit einem auf [ListServerCertificatesRequest](#). Sie gibt [ListServerCertificatesResponse](#) zurück.

Rufen Sie die `serverCertificateMetadataList` Methode des zurückgegebenen `ListServerCertificateResponse` Objekts auf, um eine Liste der [ServerCertificateMetadata](#) Objekte abzurufen, mit denen Sie Informationen zu den einzelnen Zertifikaten abrufen können.

Die Ergebnisse sind evtl. gekürzt. Wenn die `isTruncated`-Methode des `ListServerCertificateResponse`-Objekts `true` zurückgibt, rufen Sie die `marker`-Methode des `ListServerCertificatesResponse`-Objekts auf und verwenden den Marker, um eine neue Anforderung zu erstellen. Verwenden Sie die neue Anforderung, um `listServerCertificates` erneut aufzurufen, um die nächsten Teilergebnisse zu erhalten.

### Importe

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListServerCertificatesRequest;
import software.amazon.awssdk.services.iam.model.ListServerCertificatesResponse;
import software.amazon.awssdk.services.iam.model.ServerCertificateMetadata;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

### Code

```
public static void listCertificates(IamClient iam) {

    try {
        boolean done = false;
        String newMarker = null;

        while(!done) {
            ListServerCertificatesResponse response;
```



```
    if (newMarker == null) {
        ListServerCertificatesRequest request =
            ListServerCertificatesRequest.builder().build();
        response = iam.listServerCertificates(request);
    } else {
        ListServerCertificatesRequest request =
            ListServerCertificatesRequest.builder()
                .marker(newMarker).build();
        response = iam.listServerCertificates(request);
    }

    for(ServerCertificateMetadata metadata :
        response.serverCertificateMetadataList()) {
        System.out.printf("Retrieved server certificate %s",
            metadata.serverCertificateName());
    }

    if(!response.isTruncated()) {
        done = true;
    } else {
        newMarker = response.marker();
    }
}

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Aktualisieren eines Serverzertifikats

Sie können den Namen oder Pfad eines Serverzertifikats aktualisieren, indem Sie die Methode `updateServerCertificate` aufrufen. Es benötigt einen [UpdateServerCertificateRequest](#) Objektsatz mit dem aktuellen Namen des Serverzertifikats und entweder einem neuen Namen oder einem neuen Pfad.

### Importe

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateServerCertificateRequest;
import software.amazon.awssdk.services.iam.model.UpdateServerCertificateResponse;
```

## Code

```
public static void updateCertificate(IamClient iam, String curName, String newName)
{
    try {
        UpdateServerCertificateRequest request =
            UpdateServerCertificateRequest.builder()
                .serverCertificateName(curName)
                .newServerCertificateName(newName)
                .build();

        UpdateServerCertificateResponse response =
            iam.updateServerCertificate(request);

        System.out.printf("Successfully updated server certificate to name %s",
            newName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Löschen eines Serverzertifikats

Um ein Serverzertifikat zu löschen, rufen Sie die Methode `IamClient` des `deleteServerCertificate` mit einem auf, der den Namen des Zertifikats [DeleteServerCertificateRequest](#) enthält.

## Importe

```
import software.amazon.awssdk.services.iam.model.DeleteServerCertificateRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

```
import software.amazon.awssdk.services.iam.model.IamException;
```

## Code

```
public static void deleteCert(IamClient iam,String certName ) {

    try {
        DeleteServerCertificateRequest request =
            DeleteServerCertificateRequest.builder()
                .serverCertificateName(certName)
                .build();

        iam.deleteServerCertificate(request);
        System.out.println("Successfully deleted server certificate " +
            certName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Weitere Informationen

- [Arbeiten mit Serverzertifikaten](#) im IAM -Benutzerhandbuch
- [GetServerCertificate](#) in der IAM API-Referenz zu
- [ListServerCertificates](#) in der IAM API-Referenz zu
- [UpdateServerCertificate](#) in der IAM API-Referenz zu
- [DeleteServerCertificate](#) in der IAM API-Referenz zu
- [AWS Certificate Manager Benutzerhandbuch](#)

## Arbeiten mit Kinesis

Dieser Abschnitt enthält Beispiele für die Programmierung [Amazon Kinesis](#) mit AWS SDK for Java 2.x.

Weitere Informationen zu Kinesis finden Sie im [Amazon Kinesis -Entwicklerhandbuch](#).

Die folgenden Beispiele enthalten nur den Code, der zur Demonstration jeder Technik nötig ist. Der [vollständige Beispielcode ist auf verfügbar GitHub](#). Von dort aus können Sie eine einzelne Quelldatei herunterladen oder das Repository klonen, um alle Beispiele lokal zu erstellen und auszuführen.

## Themen

- [Abonnieren von Amazon Kinesis Data Streams](#)

## Abonnieren von Amazon Kinesis Data Streams

Die folgenden Beispiele zeigen Ihnen, wie Sie mit `subscribeToShard` dieser Methode Daten aus Amazon Kinesis Datenströmen abrufen und verarbeiten. Kinesis Data Streams verwendet jetzt die erweiterte Fanout-Funktion und eine HTTP/2-Datenabruf-API mit niedriger Latenz, was es Entwicklern erleichtert, mehrere Hochleistungsanwendungen mit niedriger Latenz auf demselben Datenstream auszuführen. Kinesis

## Einrichten

Erstellen Sie zunächst einen asynchronen Client und ein Objekt `Kinesis`. [SubscribeToShardRequest](#) Diese Objekte werden in jedem der folgenden Beispiele zum Abonnieren von Kinesis-Ereignissen verwendet.

## Importe

```
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.atomic.AtomicInteger;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisAsyncClient;
import software.amazon.awssdk.services.kinesis.model.ShardIteratorType;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEvent;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEventStream;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardRequest;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponse;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponseHandler;
```

## Code

```
Region region = Region.US_EAST_1;
```

```

KinesisAsyncClient client = KinesisAsyncClient.builder()
    .region(region)
    .build();

SubscribeToShardRequest request = SubscribeToShardRequest.builder()
    .consumerARN(CONSUMER_ARN)
    .shardId("arn:aws:kinesis:us-east-1:111122223333:stream/
StockTradeStream")
    .startingPosition(s -> s.type(ShardIteratorType.LATEST)).build();

```

## Verwenden Sie die Builder-Schnittstelle

Sie können die `builder` Methode verwenden, um die Erstellung von `SubscribeToShardResponseHandler` zu vereinfachen.

Mit dem Builder können Sie jeden Lebenszyklusrückruf mit einem Methodenaufruf versehen, anstatt die vollständige Schnittstelle zu implementieren.

### Code

```

private static CompletableFuture<Void> responseHandlerBuilder(KinesisAsyncClient
client, SubscribeToShardRequest request) {
    SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
    .onComplete(() -> System.out.println("All records stream
successfully"))
    // Must supply some type of subscriber
    .subscriber(e -> System.out.println("Received event - " + e))
    .build();
    return client.subscribeToShard(request, responseHandler);
}

```

Um mehr Kontrolle über den Herausgeber zu erhalten, können Sie ihn mit der Methode `publisherTransformer` anzupassen.

### Code

```

private static CompletableFuture<Void>
responseHandlerBuilderPublisherTransformer(KinesisAsyncClient client,
SubscribeToShardRequest request) {

```

```

        SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
            .builder()
            .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
            .publisherTransformer(p -> p.filter(e -> e instanceof
SubscribeToShardEvent).limit(100))
            .subscriber(e -> System.out.println("Received event - " + e))
            .build();
        return client.subscribeToShard(request, responseHandler);
    }

```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Verwenden Sie einen benutzerdefinierten Antworthandler

Implementieren Sie die `SubscribeToShardResponseHandler` Schnittstelle, um die vollständige Kontrolle über den Abonnenten und den Herausgeber zu erhalten.

In diesem Beispiel implementieren Sie die Methode `onEventStream`, die Vollzugriff auf den Herausgeber ermöglicht. Hier wird veranschaulicht, wie der Herausgeber für durch den Abonnenten zu druckende Ereignisdatensätze transformiert wird.

### Code

```

private static CompletableFuture<Void>
responseHandlerBuilderClassic(KinesisAsyncClient client, SubscribeToShardRequest
request) {
    SubscribeToShardResponseHandler responseHandler = new
SubscribeToShardResponseHandler() {

        @Override
        public void responseReceived(SubscribeToShardResponse response) {
            System.out.println("Receieved initial response");
        }

        @Override
        public void onEventStream(SdkPublisher<SubscribeToShardEventStream>
publisher) {
            publisher
                // Filter to only SubscribeToShardEvents
                .filter(SubscribeToShardEvent.class)
                // Flat map into a publisher of just records

```

```

        .flatMapIterable(SubscribeToShardEvent::records)
        // Limit to 1000 total records
        .limit(1000)
        // Batch records into lists of 25
        .buffer(25)
        // Print out each record batch
        .subscribe(batch -> System.out.println("Record Batch - " +
batch));
    }

    @Override
    public void complete() {
        System.out.println("All records stream successfully");
    }

    @Override
    public void exceptionOccurred(Throwable throwable) {
        System.err.println("Error during stream - " + throwable.getMessage());
    }
};
return client.subscribeToShard(request, responseHandler);
}

```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Verwenden Sie die Besucherschnittstelle

Sie können mithilfe eines [Visitor](#)-Objekts bestimmte Ereignisse abonnieren, an denen Sie interessiert sind.

### Code

```

private static CompletableFuture<Void>
responseHandlerBuilderVisitorBuilder(KinesisAsyncClient client,
SubscribeToShardRequest request) {
    SubscribeToShardResponseHandler.Visitor visitor =
SubscribeToShardResponseHandler.Visitor
        .builder()
        .onSubscribeToShardEvent(e -> System.out.println("Received subscribe to
shard event " + e))
        .build();
    SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler

```

```
        .builder()
        .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
        .subscriber(visitor)
        .build();
    return client.subscribeToShard(request, responseHandler);
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Verwenden Sie einen benutzerdefinierten Abonnenten

Sie können auch Ihren eigenen benutzerdefinierten Abonnenten implementieren, um den Stream zu abonnieren.

Dieser Codeausschnitt zeigt einen Beispiel für einen Abonnenten.

### Code

```
private static class MySubscriber implements
Subscriber<SubscribeToShardEventStream> {

    private Subscription subscription;
    private AtomicInteger eventCount = new AtomicInteger(0);

    @Override
    public void onSubscribe(Subscription subscription) {
        this.subscription = subscription;
        this.subscription.request(1);
    }

    @Override
    public void onNext(SubscribeToShardEventStream shardSubscriptionEventStream) {
        System.out.println("Received event " + shardSubscriptionEventStream);
        if (eventCount.incrementAndGet() >= 100) {
            // You can cancel the subscription at any time if you wish to stop
receiving events.
            subscription.cancel();
        }
        subscription.request(1);
    }

    @Override
    public void onError(Throwable throwable) {
```



```
        System.err.println("Error occurred while stream - " +
throwable.getMessage());
    }

    @Override
    public void onComplete() {
        System.out.println("Finished streaming all events");
    }
}
```

Sie können den benutzerdefinierten Abonnenten an die `subscribe` Methode übergeben, wie im folgenden Codeausschnitt gezeigt.

### Code

```
private static CompletableFuture<Void>
responseHandlerBuilderSubscriber(KinesisAsyncClient client, SubscribeToShardRequest
request) {
    SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
        .builder()
        .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
        .subscriber(MySubscriber::new)
        .build();
    return client.subscribeToShard(request, responseHandler);
}
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Schreiben Sie Datensätze in einen Kinesis Datenstrom

Sie können das [KinesisClient](#) Objekt verwenden, um Datensätze in einen Kinesis Datenstrom zu schreiben, indem Sie die `putRecords` Methode verwenden. Um diese Methode erfolgreich aufzurufen, erstellen Sie ein [PutRecordsRequest](#) Objekt. Sie übergeben den Namen des Datenstroms an die `streamName` Methode. Darüber hinaus müssen Sie die Daten mit der Methode `putRecords` übergeben (wie im folgenden Codebeispiel gezeigt).

### Importe

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;
```

Beachten Sie im folgenden Java-Codebeispiel, dass das `StockTrade` Objekt als Daten verwendet wird, um in den Kinesis Datenstrom zu schreiben. Bevor Sie dieses Beispiel ausführen, stellen Sie sicher, dass Sie den Datenstream erstellt haben.

## Code

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StockTradesWriter {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <streamName>

                Where:
                streamName - The Amazon Kinesis data stream to which records are
                written (for example, StockTradeStream)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String streamName = args[0];
    Region region = Region.US_EAST_1;
    KinesisClient kinesisClient = KinesisClient.builder()
        .region(region)
        .build();

    // Ensure that the Kinesis Stream is valid.
    validateStream(kinesisClient, streamName);
    setStockData(kinesisClient, streamName);
    kinesisClient.close();
}

public static void setStockData(KinesisClient kinesisClient, String streamName) {
    try {
        // Repeatedly send stock trades with a 100 milliseconds wait in between.
        StockTradeGenerator stockTradeGenerator = new StockTradeGenerator();

        // Put in 50 Records for this example.
        int index = 50;
        for (int x = 0; x < index; x++) {
            StockTrade trade = stockTradeGenerator.getRandomTrade();
            sendStockTrade(trade, kinesisClient, streamName);
            Thread.sleep(100);
        }

    } catch (KinesisException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done");
}

private static void sendStockTrade(StockTrade trade, KinesisClient kinesisClient,
    String streamName) {
    byte[] bytes = trade.toJsonAsBytes();

    // The bytes could be null if there is an issue with the JSON serialization by
    // the Jackson JSON library.
    if (bytes == null) {
        System.out.println("Could not get JSON bytes for stock trade");
        return;
    }
}
```

```
System.out.println("Putting trade: " + trade);
PutRecordRequest request = PutRecordRequest.builder()
    .partitionKey(trade.getTickerSymbol()) // We use the ticker symbol as
the partition key, explained in
// the Supplemental Information
section below.
    .streamName(streamName)
    .data(SdkBytes.fromByteArray(bytes))
    .build();

try {
    kinesisClient.putRecord(request);
} catch (KinesisException e) {
    System.err.println(e.getMessage());
}

private static void validateStream(KinesisClient kinesisClient, String streamName)
{
    try {
        DescribeStreamRequest describeStreamRequest =
DescribeStreamRequest.builder()
            .streamName(streamName)
            .build();

        DescribeStreamResponse describeStreamResponse =
kinesisClient.describeStream(describeStreamRequest);

        if (!
describeStreamResponse.streamDescription().streamStatus().toString().equals("ACTIVE"))
        {
            System.err.println("Stream " + streamName + " is not active. Please
wait a few moments and try again.");
            System.exit(1);
        }

    } catch (KinesisException e) {
        System.err.println("Error found while describing the stream " +
streamName);
        System.err.println(e);
        System.exit(1);
    }
}
```

```
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Verwenden Sie eine Bibliothek eines Drittanbieters

Sie können andere Drittanbieter-Bibliotheken verwenden, anstatt eine benutzerdefinierten Abonnenten zu implementieren. In diesem Beispiel wird die Verwendung der RxJava Implementierung demonstriert. Sie können jedoch jede Bibliothek verwenden, die die Reactive Streams-Schnittstellen implementiert. Weitere Informationen zu dieser Bibliothek finden Sie [auf der RxJava Wiki-Seite auf Github](#).

Um die Bibliothek zu verwenden, fügen Sie sie als Abhängigkeit hinzu. Bei Einsatz von Maven zeigt das Beispiel den zu verwendenden POM-Ausschnitt.

### POM-Eintrag

```
<dependency>  
  <groupId>io.reactivex.rxjava2</groupId>  
  <artifactId>rxjava</artifactId>  
  <version>2.1.14</version>  
</dependency>
```

### Importe

```
import java.net.URI;  
import java.util.concurrent.CompletableFuture;  
  
import io.reactivex.Flowable;  
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;  
import software.amazon.awssdk.core.async.SdkPublisher;  
import software.amazon.awssdk.http.Protocol;  
import software.amazon.awssdk.http.SdkHttpConfigurationOption;  
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.kinesis.KinesisAsyncClient;  
import software.amazon.awssdk.services.kinesis.model.ShardIteratorType;  
import software.amazon.awssdk.services.kinesis.model.StartingPosition;  
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEvent;  
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardRequest;  
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponseHandler;
```

```
import software.amazon.awssdk.utils.AttributeMap;
```

In diesem Beispiel wird die `onEventStream` Lifecycle-Methode verwendet `RxJava`. Dadurch haben Sie Vollzugriff auf den Herausgeber, der zum Erstellen einer `Rx Flowable` verwendet werden kann.

## Code

```
SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
    .onEventStream(p -> Flowable.fromPublisher(p)
        .ofType(SubscribeToShardEvent.class)

.flatMapIterable(SubscribeToShardEvent::records)
        .limit(1000)
        .buffer(25)
        .subscribe(e -> System.out.println("Record
batch = " + e)))
    .build();
```

Sie können bei dem Herausgeber `Flowable` auch die Methode `publisherTransformer` verwenden. Sie müssen den `Flowable` Herausgeber an eine anpassen `SdkPublisher`, wie im folgenden Beispiel gezeigt.

## Code

```
SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
    .publisherTransformer(p ->
SdkPublisher.adapt(Flowable.fromPublisher(p).limit(100)))
    .build();
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Weitere Informationen

- [SubscribeToShardEvent](#) in der Amazon Kinesis API-Referenz

- [SubscribeToShard](#) in der Amazon Kinesis API-Referenz

## Aufrufen, Auflisten und Löschen AWS Lambda von Funktionen

Dieser Abschnitt enthält Beispiele für die Programmierung mit dem Lambda Service-Client unter Verwendung von AWS SDK for Java 2.x.

### Themen

- [Aufrufen von einer Lambda-Funktion](#)
- [Auflisten von Lambda-Funktionen](#)
- [Löschen einer Lambda-Funktion](#)

## Aufrufen von einer Lambda-Funktion

Sie können eine Lambda Funktion aufrufen, indem Sie ein [LambdaClient](#) Objekt erstellen und dessen `invoke` Methode aufrufen. Erstellen Sie ein [InvokeRequest](#) Objekt, um zusätzliche Informationen wie den Funktionsnamen und die Nutzlast anzugeben, die an die Lambda Funktion übergeben werden soll. Funktionsnamen werden als `arn:aws:lambda:us-east-1:123456789012:function:HelloFunction` angezeigt. Sie können den Wert abrufen, indem Sie sich die Funktion in der ansehen AWS Management Console.

Um Nutzlastdaten an eine Funktion zu übergeben, erstellen Sie ein [SdkBytes](#) Objekt, das Informationen enthält. Beachten Sie beispielsweise im folgenden Codebeispiel die an die Lambda - Funktion übergebenen JSON-Daten.

### Importe

```
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.InvokeRequest;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.lambda.model.InvokeResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;
```

### Code

Das folgende Codebeispiel zeigt, wie eine Lambda Funktion aufgerufen wird.

```
public static void invokeFunction(LambdaClient awsLambda, String functionName) {

    InvokeResponse res = null ;
    try {
        //Need a SdkBytes instance for the payload
        String json = "{\"Hello \": \"Paris\"}";
        SdkBytes payload = SdkBytes.fromUtf8String(json) ;

        //Setup an InvokeRequest
        InvokeRequest request = InvokeRequest.builder()
            .functionName(functionName)
            .payload(payload)
            .build();

        res = awsLambda.invoke(request);
        String value = res.payload().asUtf8String() ;
        System.out.println(value);

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Auflisten von Lambda-Funktionen

Erstellen Sie ein [Lambda Client](#) Objekt und rufen Sie dessen `listFunctions` Methode auf. Diese Methode gibt ein [-ListFunctionsResponse](#) Objekt zurück. Sie können die `functions` Methode dieses Objekts aufrufen, um eine Liste von [FunctionConfiguration](#) Objekten zurückzugeben. Sie können die Liste durchlaufen, um Informationen über die Funktionen abzurufen. Das folgende Java-Codebeispiel zeigt beispielsweise, wie die einzelnen Funktionsnamen abgerufen werden.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.ListFunctionsResponse;
import software.amazon.awssdk.services.lambda.model.FunctionConfiguration;
```



```
import java.util.List;
```

## Code

Das folgende Java-Codebeispiel veranschaulicht, wie eine Liste von Funktionsnamen abgerufen wird.

```
public static void listFunctions(LambdaClient awsLambda) {  
  
    try {  
        ListFunctionsResponse functionResult = awsLambda.listFunctions();  
        List<FunctionConfiguration> list = functionResult.functions();  
  
        for (FunctionConfiguration config: list) {  
            System.out.println("The function name is "+config.functionName());  
        }  
  
    } catch (LambdaException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Löschen einer Lambda-Funktion

Erstellen Sie ein [LambdaClient](#) Objekt und rufen Sie dessen `deleteFunction` Methode auf. Erstellen Sie ein [DeleteFunctionRequest](#) Objekt und übergeben Sie es an die `deleteFunction` Methode. Dieses Objekt enthält Informationen wie den Namen der zu löschenden Funktion. Funktionsnamen werden als `arn:aws:lambda:us-east-1:123456789012:function:HelloFunction` angezeigt. Sie können den Wert abrufen, indem Sie sich die Funktion in der ansehen AWS Management Console.

### Importe

```
import software.amazon.awssdk.services.lambda.LambdaClient;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.lambda.model.DeleteFunctionRequest;  
import software.amazon.awssdk.services.lambda.model.LambdaException;
```

## Code

Der folgende Java-Code zeigt, wie eine Lambda Funktion gelöscht wird.

```
public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName ) {
    try {
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()
            .functionName(functionName)
            .build();

        awsLambda.deleteFunction(request);
        System.out.println("The "+functionName +" function was deleted");

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Arbeiten mit Amazon S3

Dieser Abschnitt enthält Beispiele für die Programmierung mit [Amazon Simple Storage Service \(S3\)](#) unter Verwendung der AWS SDK for Java 2.x.

Die folgenden Beispiele enthalten nur den Code, der zur Demonstration jeder Technik nötig ist. Der [vollständige Beispielcode ist auf verfügbar GitHub](#). Von dort aus können Sie eine einzelne Quelldatei herunterladen oder das Repository klonen, um alle Beispiele lokal zu erstellen und auszuführen.

### Note

Ab Version 2.18.x und höher AWS SDK for Java 2.x verwendet die Adressierung im virtuellen Host-Stil, wenn eine Endpunktüberschreibung eingeschlossen wird. Dies gilt solange der Bucket-Name eine gültige DNS-Bezeichnung ist.

Rufen Sie die [forcePathStyle](#) Methode mit `true` in Ihrem Client Builder auf, um den Client zu zwingen, die Adressierung im Pfadformat für Buckets zu verwenden.

Das folgende Beispiel zeigt einen Service-Client, der mit einer Endpunktüberschreibung konfiguriert ist und die Adressierung im Pfadformat verwendet.

```
S3Client client = S3Client.builder()
    .region(Region.US_WEST_2)
```

```
west-2.amazonaws.com"))  
        .endpointOverride(URI.create("https://s3.us-  
        .forcePathStyle(true)  
        .build();
```

## Verwenden von Zugriffspunkten oder Multi-Regions-Zugriffspunkten

Nachdem [Amazon S3-Zugriffspunkte](#) oder [Multi-Regions-Zugriffspunkte](#) eingerichtet wurden, können Sie Objektmethoden wie `putObject` und aufrufen `getObject` und die Zugriffspunkt-ID anstelle eines Bucket-Namens angeben.

Wenn beispielsweise eine Zugriffspunkt-ARN-ID lautet `arn:aws:s3:us-west-2:123456789012:accesspoint/test`, können Sie den folgenden Ausschnitt verwenden, um die Methode aufzurufen `putObject`.

```
Path path = Paths.get(URI.create("file:///temp/file.txt"));  
  
s3Client.putObject(builder -> builder  
                .key("myKey")  
                .bucket("arn:aws:s3:us-west-2:123456789012:accesspoint/test")  
                , path);
```

Anstelle der ARN-Zeichenfolge können Sie auch den [Alias im Bucket-Stil](#) des Zugriffspunkts für den `bucket` Parameter verwenden.

Um Multi-Region Access Point zu verwenden, ersetzen Sie den `bucket` Parameter durch den ARN des Multi-Region Access Points, der das folgende Format hat.

```
arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias
```

Fügen Sie die folgende Maven-Abhängigkeit hinzu, um mit Multi-Region Access Points unter Verwendung des SDK for Java zu arbeiten. Suchen Sie in Maven Central nach der [neuesten Version](#)

```
<dependency>  
  <groupId>software.amazon.awssdk</groupId>  
  <artifactId>auth-crt</artifactId>  
  <version>VERSION</version>  
</dependency>
```

## Themen

- [Amazon S3Buckets erstellen, auflisten und löschen](#)
- [Mit Amazon S3 Objekten arbeiten](#)
- [Mit Amazon S3 vorsignierten URLs arbeiten](#)
- [Regionsübergreifender Zugriff für Amazon S3](#)
- [Amazon S3 S3-Prüfsummen mit 3](#)
- [Verwenden Sie einen leistungsstarken S3-Client: AWS CRT-basierter S3-Client](#)
- [Übertragen von Dateien und Verzeichnissen mit dem Amazon S3 Transfer Manager](#)

## Amazon S3Buckets erstellen, auflisten und löschen

Jedes Objekt (Datei) in Amazon S3 muss sich in einem Bucket befinden. Ein Bucket stellt eine Sammlung von Objekten (Container) dar. Jeder Bucket muss einen eindeutigen Schlüssel (Name) haben. Ausführliche Informationen zu Buckets und ihrer Konfiguration finden Sie unter [Working with Amazon S3 Buckets](#) im Amazon Simple Storage Service Benutzerhandbuch.

### Note

#### Bewährte Methode

Wir empfehlen, dass Sie die [AbortIncompleteMultipartUpload](#)Lebenszyklusregel für Ihre Amazon S3 Buckets aktivieren.

Diese Regel weist Amazon S3 an, mehrteilige Uploads abubrechen, die nicht innerhalb einer bestimmten Anzahl von Tagen nach dem Start abgeschlossen werden. Wenn die festgelegte Höchstdauer überschritten wird, bricht Amazon S3 den Upload ab und löscht dann die unvollständigen Upload-Daten.

Weitere Informationen finden Sie unter [Lebenszykluskonfiguration für einen Bucket mit Versionierung](#) im Amazon Simple Storage Service Benutzerhandbuch.

### Note

Bei diesen Codefragmenten wird davon ausgegangen, dass Sie das Material in den Grundlagen verstehen und AWS Standardanmeldedaten anhand der Informationen in konfiguriert haben. [the section called “Einrichten des Single-Sign-On-Zugriffs für das SDK”](#)

## Erstellen eines -Buckets

Erstellen Sie einen Bucket [CreateBucketRequest](#) und geben Sie einen Namen an. Übergeben Sie ihn an die Methode des `S3Client`. `createBucket` Verwenden Sie den `S3Client`, um zusätzliche Operationen wie das Auflisten oder Löschen von Buckets, wie in späteren Beispielen gezeigt, durchzuführen.

### Importe

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

### Code

Erstellen Sie zuerst einen `S3Client`.

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

Stellen Sie eine Anfrage zum Erstellen eines Buckets.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
```

```
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3BucketOps {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        String bucket = "bucket" + System.currentTimeMillis();
        System.out.println(bucket);
        createBucket(s3, bucket);
        performOperations(s3, bucket);
    }

    // Create a bucket by using a S3Waiter object
    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                .bucket(bucketName)
                .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Buckets auflisten

Baue ein [ListBucketsRequest](#). Verwenden Sie die `listBuckets` Methode des `S3Client`, um die Liste der Buckets abzurufen. Wenn die Anfrage erfolgreich ist, wird `a` zurückgegeben.

[ListBucketsResponse](#) Verwenden Sie dieses Antwortobjekt zum Abrufen der Liste der Buckets.

### Importe

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

### Code

Erstellen Sie zuerst einen `S3Client`.

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

Stellen Sie eine Anfrage zum Auflisten der Buckets.

```
// List buckets
ListBucketsRequest listBucketsRequest = ListBucketsRequest.builder().build();
```

```
ListBucketsResponse listBucketsResponse = s3.listBuckets(listBucketsRequest);
listBucketsResponse.buckets().stream().forEach(x ->
System.out.println(x.name()));
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Löschen eines -Buckets

Bevor Sie einen Amazon S3-Bucket löschen können, müssen Sie sicherstellen, dass der Bucket leer ist. Andernfalls gibt der Service einen Fehler zurück. Wenn Sie einen [versionierten Bucket](#) nutzen, müssen Sie außerdem alle versionierten Objekte löschen, die sich im Bucket befinden.

### Themen

- [Objekte in einem Bucket löschen](#)
- [Einen leeren Bucket löschen](#)

### Objekte in einem Bucket löschen

Erstellen Sie eine [ListObjectsV2Request](#) und verwenden Sie die `listObjects` Methode des `S3Client`, um die Liste der Objekte im Bucket abzurufen. Anschließend verwenden Sie die `deleteObject`-Methode für jedes Objekt, um es zu löschen.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
```

### Code

Erstellen Sie zuerst einen `S3Client`.

```
ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
Region region = Region.US_EAST_1;
```



```
S3Client s3 = S3Client.builder()
    .region(region)
    .credentialsProvider(credentialsProvider)
    .build();
```

Löschen Sie alle Objekte im Bucket.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3BucketDeletion {
    public static void main(String[] args) throws Exception {
        final String usage = ""

            Usage:
                <bucket>

            Where:
                bucket - The bucket to delete (for example, bucket1).\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
```

```

        .region(region)
        .build();

deleteObjectsInBucket(s3, bucket);
s3.close();
}

public static void deleteObjectsInBucket(S3Client s3, String bucket) {
    try {
        // To delete a bucket, all the objects in the bucket must be deleted first.
        ListObjectsV2Request listObjectsV2Request = ListObjectsV2Request.builder()
            .bucket(bucket)
            .build();
        ListObjectsV2Response listObjectsV2Response;

        do {
            listObjectsV2Response = s3.listObjectsV2(listObjectsV2Request);
            for (S3Object s3Object : listObjectsV2Response.contents()) {
                DeleteObjectRequest request = DeleteObjectRequest.builder()
                    .bucket(bucket)
                    .key(s3Object.key())
                    .build();
                s3.deleteObject(request);
            }
        } while (listObjectsV2Response.isTruncated());
        DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder().bucket(bucket).build();
s3.deleteBucket(deleteBucketRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Einen leeren Bucket löschen

Erstellen Sie eine [DeleteBucketRequest](#) mit einem Bucket-Namen und übergeben Sie sie an die Methode des S3Client. `deleteBucket`

### Importe

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

## Code

Erstellen Sie zuerst einen S3Client.

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

Löschen Sie den Bucket.

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucket)
    .build();

s3.deleteBucket(deleteBucketRequest);
s3.close();
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Mit Amazon S3 Objekten arbeiten

Ein Amazon S3-Objekt stellt eine Datei oder eine Sammlung von Daten dar. Jedes Objekt ist in einem [Bucket](#) enthalten sein.

### Note

Bewährte Methode

Wir empfehlen, dass Sie die [AbortIncompleteMultipartUpload](#) Lebenszyklusregel für Ihre Amazon S3 Buckets aktivieren.

Diese Regel weist Amazon S3 an, mehrteilige Uploads abzubrechen, die nicht innerhalb einer bestimmten Anzahl von Tagen nach dem Start abgeschlossen werden. Wenn die festgelegte Höchstdauer überschritten wird, bricht Amazon S3 den Upload ab und löscht dann die unvollständigen Upload-Daten.

Weitere Informationen finden Sie unter [Lebenszykluskonfiguration für einen Bucket mit Versionierung](#) im Amazon Simple Storage Service Benutzerhandbuch.

### Note

Bei diesen Codefragmenten wird davon ausgegangen, dass Sie das Material in den Grundlagen verstehen und AWS Standardanmeldedaten anhand der Informationen in konfiguriert haben. [the section called “Einrichten des Single-Sign-On-Zugriffs für das SDK”](#)

## Themen

- [Hochladen eines Objekts](#)
- [Hochladen von Objekten in mehreren Teilen](#)
- [Ein Objekt löschen](#)
- [List objects](#)
- [Weitere Beispiele](#)

## Hochladen eines Objekts

Erstellen Sie eine [PutObjectRequest](#) und geben Sie einen Bucket- und Schlüsselnamen an. Verwenden Sie dann die `putObject` Methode des `S3Client` mit einer [RequestBody](#), die den Objektkinhalt und das `PutObjectRequest` Objekt enthält. Der Bucket muss vorhanden sein, andernfalls gibt der Service einen Fehler zurück.

## Importe

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
```

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

## Code

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

createBucket(s3, bucketName, region);

PutObjectRequest objectRequest = PutObjectRequest.builder()
    .bucket(bucketName)
    .key(key)
    .build();

s3.putObject(objectRequest,
    RequestBody.fromByteBuffer(getRandomByteBuffer(10_000)));
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Hochladen von Objekten in mehreren Teilen

Verwenden Sie die `createMultipartUpload` Methode des `S3Client`, um eine Upload-ID zu erhalten. Anschließend verwenden Sie die `uploadPart`-Methode, um die einzelnen Teile hochzuladen. Verwenden Sie abschließend die `completeMultipartUpload` Methode des `S3Client`, um anzuweisen, dass alle hochgeladenen Teile zusammengeführt und der Upload-Vorgang abgeschlossen werden Amazon S3 soll.

### Importe

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

### Code

```
// First create a multipart upload and get the upload id
```

```
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
        String uploadId = response.uploadId();
        System.out.println(uploadId);

        // Upload all the different parts of the object
        UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)
            .partNumber(1).build();

        String etag1 = s3
            .uploadPart(uploadPartRequest1,
RequestBody.fromByteBuffer(getRandomByteBuffer(5 * mB)))
            .eTag();

        CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

        UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
            .uploadId(uploadId)
            .partNumber(2).build();

        String etag2 = s3
            .uploadPart(uploadPartRequest2,
RequestBody.fromByteBuffer(getRandomByteBuffer(3 * mB)))
            .eTag();

        CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

        // Finally call completeMultipartUpload operation to tell S3 to merge
all
        // uploaded
        // parts and finish the multipart operation.
        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
            .parts(part1, part2)
```

```
        .build();

        CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
        .bucket(bucketName)
        .key(key)
        .uploadId(uploadId)
        .multipartUpload(completedMultipartUpload)
        .build();

        s3.completeMultipartUpload(completeMultipartUploadRequest);
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Ein Objekt löschen

Erstellen Sie einen [DeleteObjectRequest](#) und geben Sie einen Bucket-Namen und einen Schlüsselnamen an. Verwenden Sie die `deleteObject` Methode des `S3Client` und übergeben Sie ihr den Namen eines Buckets und eines Objekts, das gelöscht werden soll. Der angegebene Bucket und der Objektschlüssel müssen vorhanden sein, andernfalls gibt der Service einen Fehler zurück.

### Importe

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
```



```
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

## Code

```
DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
    .bucket(bucketName)
    .key(key)
    .build();

s3.deleteObject(deleteObjectRequest);
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Kopieren eines Objekts

Erstellen Sie einen [CopyObjectRequest](#) und geben Sie einen Bucket-Namen an, in den das Objekt kopiert wird, einen URL-codierten Zeichenfolgenwert (siehe die Methode `UrlEncoder.Encode`) und den Schlüsselnamen des Objekts. Verwenden Sie die Methode von `S3Client` und übergeben Sie das Objekt. `copyObject` [CopyObjectRequest](#) Der angegebene Bucket und der Objektschlüssel müssen vorhanden sein, andernfalls gibt der Service einen Fehler zurück.

## Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

## Code

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CopyObject {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <objectKey> <fromBucket> <toBucket>

            Where:
                objectKey - The name of the object (for example, book.pdf).
                fromBucket - The S3 bucket name that contains the object (for
example, bucket1).
                toBucket - The S3 bucket to copy the object to (for example,
bucket2).

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String objectKey = args[0];
        String fromBucket = args[1];
        String toBucket = args[2];
        System.out.format("Copying object %s from bucket %s to %s\n", objectKey,
fromBucket, toBucket);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        copyBucketObject(s3, fromBucket, objectKey, toBucket);
        s3.close();
    }
}
```

```
public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .sourceBucket(fromBucket)
        .sourceKey(objectKey)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    try {
        CopyObjectResponse copyRes = s3.copyObject(copyReq);
        return copyRes.copyObjectResult().toString();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## List objects

Erstellen Sie einen [ListObjectsRequest](#) und geben Sie den Bucket-Namen an. Rufen Sie dann die `listObjects` Methode des `S3Client` auf und übergeben Sie das Objekt. `ListObjectsRequest` Diese Methode gibt zurück [ListObjectsResponse](#), das alle Objekte im Bucket enthält. Sie können die `getContent` Methode dieses Objekts aufrufen, um eine Liste von Objekten anzufordern. Sie können durch diese Liste iterieren, um die Objekte anzuzeigen, wie im folgenden Codebeispiel gezeigt.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;
```

### Code

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListObjects {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }
}
```

```
public static void listBucketObjects(S3Client s3, String bucketName) {
    try {
        ListObjectsRequest listObjects = ListObjectsRequest
            .builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.print("\n The name of the key is " + myValue.key());
            System.out.print("\n The object is " + calKb(myValue.size()) + " KBs");
            System.out.print("\n The owner is " + myValue.owner());
        }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// convert bytes to kbs.
private static long calKb(Long val) {
    return val / 1024;
}
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Weitere Beispiele

Der Abschnitt mit den [Codebeispielen](#) dieses Handbuchs enthält weitere Beispiele für die Arbeit mit Amazon S3 S3-Objekten, einschließlich des [Herunterladens eines Objekts](#).

## Mit Amazon S3 vorsignierten URLs arbeiten

Vorsignierte URLs bieten temporären Zugriff auf private S3-Objekte, ohne dass Benutzer über AWS Anmeldeinformationen oder Berechtigungen verfügen müssen.

Angenommen, Alice hat Zugriff auf ein S3-Objekt, und sie möchte den Zugriff auf dieses Objekt vorübergehend mit Bob teilen. Alice kann eine vorsignierte GET-Anfrage generieren, um sie mit Bob zu teilen, sodass er das Objekt herunterladen kann, ohne Zugriff auf Alices Anmeldeinformationen zu benötigen. Sie können vorsignierte URLs für HTTP-GET- und für HTTP-PUT-Anfragen generieren.

## Generieren Sie eine vorsignierte URL für ein Objekt und laden Sie sie dann herunter (GET-Anfrage)

Das folgende Beispiel besteht aus zwei Teilen.

- Teil 1: Alice generiert die vorsignierte URL für ein Objekt.
- Teil 2: Bob lädt das Objekt mithilfe der vorsignierten URL herunter.

### Teil 1: Generieren Sie die URL

Alice hat bereits ein Objekt in einem S3-Bucket. Sie verwendet den folgenden Code, um eine URL-Zeichenfolge zu generieren, die Bob in einer nachfolgenden GET-Anfrage verwenden kann.

### Importe

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
```

```
import java.time.Duration;
import java.util.UUID;
```

```
/* Create a pre-signed URL to download an object in a subsequent GET request. */
public String createPresignedGetUrl(String bucketName, String keyName) {
    try (S3Presigner presigner = S3Presigner.create()) {

        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        GetObjectPresignRequest presignRequest = GetObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // The URL will expire
in 10 minutes.
            .getObjectRequest(objectRequest)
            .build();

        PresignedGetObjectRequest presignedRequest =
presigner.presignGetObject(presignRequest);
        logger.info("Presigned URL: [{}]", presignedRequest.url().toString());
        logger.info("HTTP method: [{}]", presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}
```

## Teil 2: Laden Sie das Objekt herunter

Bob verwendet eine der folgenden drei Codeoptionen, um das Objekt herunterzuladen. Alternativ könnte er einen Browser verwenden, um die GET-Anfrage auszuführen.

### Benutze JDK **HttpURLConnection** (seit v1.1)

```
/* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
public byte[] useHttpURLConnectionToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //
Capture the response body to a byte array.

    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
```

```

        connection.setRequestMethod("GET");
        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            IoUtils.copy(content, byteArrayOutputStream);
        }
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}

```

## Benutze JDK `HttpClient` (seit v11)

```

/* Use the JDK HttpClient (since v11) class to do the download. */
public byte[] useHttpClientToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //
    Capture the response body to a byte array.

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpResponse<InputStream> response = httpClient.send(requestBuilder
            .uri(presignedUrl.toURI())
            .GET()
            .build(),
            HttpResponse.BodyHandlers.ofInputStream());

        IoUtils.copy(response.body(), byteArrayOutputStream);

        logger.info("HTTP response code is " + response.statusCode());

    } catch (URISyntaxException | InterruptedException | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}

```

## Verwendung `SdkHttpClient` aus dem SDK for Java

```

/* Use the AWS SDK for Java SdkHttpClient class to do the download. */

```



```
public byte[] useSdkHttpClientToPut(String presignedUrlString) {

    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //
    Capture the response body to a byte array.
    try {
        URL presignedUrl = new URL(presignedUrlString);
        SdkHttpRequest request = SdkHttpRequest.builder()
            .method(SdkHttpMethod.GET)
            .uri(presignedUrl.toURI())
            .build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
            response.responseBody().ifPresentOrElse(
                abortableInputStream -> {
                    try {
                        IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                },
                () -> logger.error("No response body."));

            logger.info("HTTP Response code is {}",
response.httpResponse().statusCode());
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) an und [testen Sie](#) es weiter GitHub.

## Generieren Sie eine vorsignierte URL für einen Upload und laden Sie dann eine Datei hoch (PUT-Anfrage)

Das folgende Beispiel besteht aus zwei Teilen.

- Teil 1: Alice generiert die vorsignierte URL, um ein Objekt hochzuladen.
- Teil 2: Bob lädt eine Datei mithilfe der vorsignierten URL hoch.

### Teil 1: Generieren Sie die URL

Alice hat bereits einen S3-Bucket. Sie verwendet den folgenden Code, um eine URL-Zeichenfolge zu generieren, die Bob in einer nachfolgenden PUT-Anfrage verwenden kann.

### Importe

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
```

```
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

```
/* Create a presigned URL to use in a subsequent PUT request */
public String createPresignedUrl(String bucketName, String keyName, Map<String,
String> metadata) {
    try (S3Presigner presigner = S3Presigner.create()) {

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .metadata(metadata)
            .build();

        PutObjectPresignRequest presignRequest = PutObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // The URL expires in
10 minutes.

            .putObjectRequest(objectRequest)
            .build();

        PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
        String myURL = presignedRequest.url().toString();
        logger.info("Presigned URL to upload a file to: [{}]", myURL);
        logger.info("HTTP method: [{}]", presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}
```

Teil 2: Laden Sie ein Dateiojekt hoch

Bob verwendet eine der folgenden drei Codeoptionen, um eine Datei hochzuladen.

Benutze JDK **HttpURLConnection** (seit v1.1)

```
/* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
```

```

public void useHttpURLConnectionToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setDoOutput(true);
        metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-meta-" + k,
v));
        connection.setRequestMethod("PUT");
        OutputStream out = connection.getOutputStream();

        try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
            FileChannel inChannel = file.getChannel()) {
            ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is 8k

            while (inChannel.read(buffer) > 0) {
                buffer.flip();
                for (int i = 0; i < buffer.limit(); i++) {
                    out.write(buffer.get());
                }
                buffer.clear();
            }
        } catch (IOException e) {
            logger.error(e.getMessage(), e);
        }

        out.close();
        connection.getResponseCode();
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
}

```

## Benutze JDK **HttpClient** (seit v11)

```

/* Use the JDK HttpClient (since v11) class to do the upload. */
public void useHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

```

```

HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

HttpClient httpClient = HttpClient.newHttpClient();
try {
    final HttpResponse<Void> response = httpClient.send(requestBuilder
        .uri(new URL(presignedUrlString).toURI())
        .PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI()))
            .build(),
            HttpResponse.BodyHandlers.discarding());

    logger.info("HTTP response code is " + response.statusCode());

} catch (URISyntaxException | InterruptedException | IOException e) {
    logger.error(e.getMessage(), e);
}
}

```

## Verwendung **SdkHttpClient** aus dem SDK for Java

```

/* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    try {
        URL presignedUrl = new URL(presignedUrlString);

        SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
            .method(SdkHttpMethod.PUT)
            .uri(presignedUrl.toURI());

        // Add headers
        metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" + k, v));
        // Finish building the request.
        SdkHttpRequest request = requestBuilder.build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
            .build();

```

```
        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
            logger.info("Response code: {}", response.httpResponse().statusCode());
        }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) an und [testen Sie](#) es weiter GitHub.

## Regionsübergreifender Zugriff für Amazon S3

Wenn Sie mit Amazon Simple Storage Service (Amazon S3) -Buckets arbeiten, kennen Sie normalerweise das AWS-Region für den Bucket. Die Region, mit der Sie arbeiten, wird bei der Erstellung des S3-Clients festgelegt.

Manchmal müssen Sie jedoch möglicherweise mit einem bestimmten Bucket arbeiten, wissen aber nicht, ob sich dieser in derselben Region befindet, die für den S3-Client festgelegt ist.

Anstatt mehr Aufrufe zu tätigen, um die Bucket-Region zu ermitteln, können Sie das SDK verwenden, um den Zugriff auf S3-Buckets in verschiedenen Regionen zu ermöglichen.

### Setup

Die Support für den regionsübergreifenden Zugriff wurde mit 2.20.111 der Version des SDK verfügbar. Verwenden Sie diese oder eine neuere Version in Ihrer Maven-Build-Datei für die s3 Abhängigkeit, wie im folgenden Codeausschnitt gezeigt.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>s3</artifactId>
  <version>2.20.111</version>
</dependency>
```

Wenn Sie als Nächstes Ihren S3-Client erstellen, aktivieren Sie den regionsübergreifenden Zugriff, wie im Snippet gezeigt. Standardmäßig ist der Zugriff nicht aktiviert.

```
S3AsyncClient client = S3AsyncClient.builder()
    .crossRegionAccessEnabled(true)
```

```
.build();
```

## Wie das SDK regionsübergreifenden Zugriff bietet

Wenn Sie in einer Anfrage auf einen vorhandenen Bucket verweisen, z. B. wenn Sie die `putObject` Methode verwenden, initiiert das SDK eine Anfrage an die für den Client konfigurierte Region.

Wenn der Bucket in dieser bestimmten Region nicht existiert, umfasst die Fehlerantwort die tatsächliche Region, in der sich der Bucket befindet. Das SDK verwendet dann in einer zweiten Anfrage die richtige Region.

Um future Anfragen an denselben Bucket zu optimieren, speichert das SDK diese Regionszuweisung im Client zwischen.

## Überlegungen

Wenn Sie den regionsübergreifenden Bucket-Zugriff aktivieren, beachten Sie, dass der erste API-Aufruf zu einer erhöhten Latenz führen kann, wenn sich der Bucket nicht in der konfigurierten Region des Clients befindet. Nachfolgende Aufrufe profitieren jedoch von zwischengespeicherten Regionsinformationen, was zu einer verbesserten Leistung führt.

Wenn Sie den regionsübergreifenden Zugriff aktivieren, wird der Zugriff auf den Bucket nicht beeinträchtigt. Der Benutzer muss berechtigt sein, auf den Bucket in der Region zuzugreifen, in der er sich befindet.

## Amazon S3 S3-Prüfsummen mit 3

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 User Guide](#).

Amazon S3 unterstützt derzeit vier Prüfsummenalgorithmen: SHA-1, SHA-256, CRC-32 und CRC-32C. Sie haben die Flexibilität, den Algorithmus auszuwählen, der Ihren Anforderungen am besten entspricht, und das SDK die Prüfsumme berechnen zu lassen. Alternativ können Sie ihren eigenen vorberechneten Prüfsummenwert angeben, indem Sie einen der vier unterstützten Algorithmen verwenden.

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

## Hochladen eines Objekts

Gültige Werte für den Algorithmus sind CRC32, CRC32CSHA1, und SHA256

Der folgende Codeausschnitt zeigt eine Anforderung zum Hochladen eines Objekts mit einer CRC-32-Prüfsumme. Wenn das SDK die Anfrage sendet, berechnet es die CRC-32-Prüfsumme und lädt das Objekt hoch. Amazon S3 speichert die Prüfsumme zusammen mit dem Objekt.

Wenn die vom SDK berechnete Prüfsumme nicht mit der Prüfsumme übereinstimmt, die Amazon S3 beim Empfang der Anfrage berechnet, wird ein Fehler zurückgegeben.

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 SHA-256-Prüfsumme.

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.

## Herunterladen eines Objekts

Wenn Sie die [GetObject-Methode](#) verwenden, um ein Objekt herunterzuladen, validiert das SDK automatisch die Prüfsumme, wenn `enableChecksumValidation` auf `true` gesetzt ist.

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.

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

Ein Objekt in Amazon S3 kann mehrere Prüfsummen haben, aber nur eine Prüfsumme wird beim Herunterladen validiert. Die folgende Rangfolge — basierend auf der Effizienz des Prüfsummenalgorithmus — bestimmt, welche Prüfsumme das SDK validiert:



1. CRC-32C
2. CRC-32
3. SHA-1
4. SHA-256

Wenn eine Antwort beispielsweise sowohl CRC-32- als auch SHA-256-Prüfsummen enthält, wird nur die CRC-32-Prüfsumme validiert.

## Verwenden Sie einen leistungsstarken S3-Client: AWS CRT-basierter S3-Client

Der AWS CRT-basierte S3-Client, der auf [AWSCommon Runtime \(CRT\)](#) aufbaut, ist ein alternativer asynchroner S3-Client. Es überträgt Objekte zu und von Amazon Simple Storage Service (Amazon S3) mit verbesserter Leistung und Zuverlässigkeit, indem es automatisch die [mehrteilige Upload-API](#) und [Bytebereichs-Abrufe](#) von Amazon S3 verwendet.

Der AWS CRT-basierte S3-Client verbessert die Zuverlässigkeit der Übertragung im Falle eines Netzwerkausfalls. Die Zuverlässigkeit wird verbessert, indem einzelne fehlgeschlagene Teile einer Dateiübertragung erneut versucht werden, ohne die Übertragung von vorne neu zu starten.

Darüber hinaus bietet der AWS CRT-basierte S3-Client ein erweitertes Verbindungspooling und einen DNS-Lastenausgleich (Domain Name System), wodurch auch der Durchsatz verbessert wird.

Sie können den AWS CRT-basierten S3-Client anstelle des standardmäßigen asynchronen S3-Clients des SDK verwenden und sofort von seinem verbesserten Durchsatz profitieren.

### AWSCRT-basierte Komponenten im SDK

Der AWS CRT-basierte S3-Client, der in diesem Thema beschrieben wird, und der AWS CRT-basierte HTTP-Client sind unterschiedliche Komponenten im SDK.

Der AWSCRT-basierte S3-Client ist eine Implementierung der [AsyncClientS3-Schnittstelle](#) und wird für die Arbeit mit dem Amazon S3 S3-Service verwendet. Es ist eine Alternative zur Java-basierten Implementierung der `S3AsyncClient` Schnittstelle und bietet mehrere Vorteile.

Der [AWSCRT-basierte HTTP-Client](#) ist eine Implementierung der [SdkAsyncHttpClient](#) Schnittstelle und wird für die allgemeine HTTP-Kommunikation verwendet. Es ist eine Alternative zur Netty-Implementierung der `SdkAsyncHttpClient` Schnittstelle und bietet mehrere Vorteile.

Obwohl beide Komponenten Bibliotheken aus der [AWSCommon Runtime](#) verwenden, verwendet der AWS CRT-basierte S3-Client die [3-Bibliothek](#) und unterstützt die aws-c-s [S3-Funktionen der mehrteiligen Upload-API](#). Da der AWS CRT-basierte HTTP-Client für allgemeine Zwecke vorgesehen ist, unterstützt er die API-Funktionen für mehrteilige Uploads von S3 nicht.

Fügen Sie Abhängigkeiten hinzu, um den CRT-basierten S3-Client zu verwenden AWS

Um den AWS CRT-basierten S3-Client zu verwenden, fügen Sie Ihrer Maven-Projektdatei die folgenden beiden Abhängigkeiten hinzu. Das Beispiel zeigt die Mindestversionen, die verwendet werden müssen. Suchen Sie im zentralen Maven-Repository nach den neuesten Versionen der [s3](#) - und [aws-crt-Artefakte](#).

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>s3</artifactId>
  <version>2.20.68</version>
</dependency>
<dependency>
  <groupId>software.amazon.awssdk.crt</groupId>
  <artifactId>aws-crt</artifactId>
  <version>0.21.16</version>
</dependency>
```

## Erstellen Sie eine Instanz des CRT-basierten S3-Clients AWS

Erstellen Sie eine Instanz des AWS CRT-basierten S3-Clients mit Standardeinstellungen, wie im folgenden Codeausschnitt gezeigt.

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate();
```

Verwenden Sie den CRT-Client-Builder, um den AWS Client zu konfigurieren. Sie können vom standardmäßigen asynchronen S3-Client zum AWS CRT-basierten Client wechseln, indem Sie die Builder-Methode ändern.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;

S3AsyncClient s3AsyncClient =
```

```
S3AsyncClient.crtBuilder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_WEST_2)
    .targetThroughputInGbps(20.0)
    .minimumPartSizeInBytes(8 * 1025 * 1024L)
    .build();
```

### Note

Einige Einstellungen im Standard-Builder werden derzeit möglicherweise nicht im AWS CRT-Client-Builders unterstützt. Rufen Sie den Standard Builder an, indem Sie `anrufenS3AsyncClient#builder()` aufrufen.

## Verwenden Sie den AWS CRT-basierten S3-Client

Verwenden Sie den AWS CRT-basierten S3-Client, um Amazon S3 S3-API-Operationen aufzurufen. Das folgende Beispiel zeigt die [GetObject](#)-Operationen [PutObject](#) und, die über die verfügbar sind. AWS SDK for Java

```
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.AsyncResponseTransformer;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

S3AsyncClient s3Client = S3AsyncClient.crtCreate();

// Upload a local file to Amazon S3.
PutObjectResponse putObjectResponse =
    s3Client.putObject(req -> req.bucket(<BUCKET_NAME>)
        .key(<KEY_NAME>),
        AsyncRequestBody.fromFile(Paths.get(<FILE_NAME>)))
        .join();

// Download an object from Amazon S3 to a local file.
GetObjectResponse getObjectResponse =
    s3Client.getObject(req -> req.bucket(<BUCKET_NAME>)
        .key(<KEY_NAME>),
        AsyncResponseTransformerToFile(Paths.get(<FILE_NAME>)))
        .join();
```

# Übertragen von Dateien und Verzeichnissen mit dem Amazon S3 Transfer Manager

Der Amazon S3 Transfer Manager ist ein Open-Source-Dienstprogramm für die Dateiübertragung auf hoher Ebene für die AWS SDK for Java 2.x. Verwenden Sie sie, um Dateien und Verzeichnisse zu und von Amazon Simple Storage Service (Amazon S3) zu übertragen.

Wenn der S3 Transfer Manager auf dem [AWS CRT-basierten S3-Client aufbaut](#), kann er Leistungsverbesserungen wie die [API für mehrteilige Uploads](#) und [Abrufe im Bytebereich](#) nutzen.

Mit dem S3 Transfer Manager können Sie auch den Fortschritt einer Übertragung in Echtzeit überwachen und die Übertragung zur späteren Ausführung anhalten.

## Erste Schritte

### Hinzufügen von Abhängigkeiten zu Ihrer Build-Datei

Um den S3 Transfer Manager mit verbesserter Leistung basierend auf dem AWS CRT-basierten S3-Client zu verwenden, konfigurieren Sie Ihre Build-Datei mit den folgenden Abhängigkeiten.

- Verwenden Sie Version **2.19.1** oder höher des SDK for Java 2.x.
- Fügen Sie das `s3-transfer-manager` Artefakt als Abhängigkeit hinzu.
- Fügen Sie das `aws-crt` Artefakt als Abhängigkeit in Version **0.20.3** oder höher hinzu.

Das folgende Codebeispiel zeigt, wie Sie Ihre Projektabhängigkeiten für Maven konfigurieren.

```
<project>
  <properties>
    <aws.sdk.version>2.19.1</aws.sdk.version>
  </properties>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>${aws.sdk.version}</version>
        <type>pom</type>
      </dependency>
    </dependencies>
  </dependencyManagement>
</project>
```

```
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
<dependencies>
<dependency>
<groupId>software.amazon.awssdk</groupId>
<artifactId>s3-transfer-manager</artifactId>
</dependency>
<dependency>
<groupId>software.amazon.awssdk.crt</groupId>
<artifactId>aws-crt</artifactId>
<version>0.20.3</version>
</dependency>
</dependencies>
</project>
```

Suchen Sie im zentralen Maven-Repository nach den neuesten Versionen der Artefakte [s3-transfer-manager](#) und [aws-crt](#).

## Erstellen einer Instance des S3 Transfer Manager

Der folgende Codeausschnitt zeigt, wie Sie eine [S3TransferManager](#)-Instance mit Standardeinstellungen erstellen.

```
S3TransferManager transferManager = S3TransferManager.create();
```

Das folgende Beispiel zeigt, wie Sie einen S3 Transfer Manager mit benutzerdefinierten Einstellungen konfigurieren. In diesem Beispiel wird eine [AWS CRT-basierte S3AsyncClient](#)-Instance als zugrunde liegender Client für den S3 Transfer Manager verwendet.

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .targetThroughputInGbps(20.0)
    .minimumPartSizeInBytes(8 * MB)
    .build();

S3TransferManager transferManager = S3TransferManager.builder()
    .s3Client(s3AsyncClient)
    .build();
```

**Note**

Wenn die `aws-crt` Abhängigkeit nicht in der Build-Datei enthalten ist, baut der S3 Transfer Manager auf dem asynchronen Standard-S3-Client auf, der im SDK für Java 2.x verwendet wird.

## Hochladen einer Datei in einen S3-Bucket

Das folgende Beispiel zeigt ein Beispiel für den Datei-Upload zusammen mit der optionalen Verwendung eines [LoggingTransferListener](#), das den Fortschritt des Uploads protokolliert.

Um eine Datei mit dem S3 Transfer Manager in Amazon S3 hochzuladen, übergeben Sie ein [UploadFileRequest](#) Objekt an die [uploadFile](#)-Methode `S3TransferManager` des `s`.

Das von der `uploadFile` Methode zurückgegebene [FileUpload](#) Objekt stellt den Upload-Prozess dar. Nach Abschluss der Anforderung enthält das [CompletedFileUpload](#) Objekt Informationen über den Upload.

```
public String uploadFile(S3TransferManager transferManager, String bucketName,
                        String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create())
        .source(Paths.get(filePathURI))
        .build();

    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

    CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
    return uploadResult.response().eTag();
}
```

## Importe

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
```

```
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;
```

## Herunterladen einer Datei aus einem S3-Bucket

Das folgende Beispiel zeigt ein Download-Beispiel zusammen mit der optionalen Verwendung eines [LoggingTransferListener](#), das den Fortschritt des Downloads protokolliert.

Um ein Objekt aus einem S3-Bucket mit dem S3 Transfer Manager herunterzuladen, erstellen Sie ein [DownloadFileRequest](#) Objekt und übergeben es an die [downloadFile](#)-Methode.

Das von der Methode des S3TransferManager zurückgegebene [FileDownload](#) Objekt `downloadFile` stellt die Dateiübertragung dar. Nach Abschluss des Downloads [CompletedFileDownload](#) enthält die Zugriff auf Informationen zum Download.

```
public Long downloadFile(S3TransferManager transferManager, String bucketName,
                        String key, String downloadedFileWithPath) {
    DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
        .getObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create())
        .destination(Paths.get(downloadedFileWithPath))
        .build();

    FileDownload downloadFile = transferManager.downloadFile(downloadFileRequest);

    CompletedFileDownload downloadResult = downloadFile.completionFuture().join();
    logger.info("Content length [{}]", downloadResult.response().contentLength());
    return downloadResult.response().contentLength();
}
```

## Importe

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
```

```
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;
```

## Kopieren eines Amazon S3-Objekts in einen anderen Bucket

Das folgende Beispiel zeigt, wie Sie ein Objekt mit dem S3 Transfer Manager kopieren.

Um mit dem Kopieren eines Objekts aus einem S3-Bucket in einen anderen Bucket zu beginnen, erstellen Sie eine Basis-[CopyObjectRequest](#)Instance.

Verpacken Sie als Nächstes das Basic CopyObjectRequest in eine [CopyRequest](#), die vom S3 Transfer Manager verwendet werden kann.

Das von der Methode des S3TransferManager zurückgegebene Copy Objekt copy stellt den Kopiervorgang dar. Nach Abschluss des Kopiervorgangs enthält das [CompletedCopy](#) Objekt Details zur Antwort.

```
public String copyObject(S3TransferManager transferManager, String bucketName,
    String key, String destinationBucket, String destinationKey) {
    CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
        .sourceBucket(bucketName)
        .sourceKey(key)
        .destinationBucket(destinationBucket)
        .destinationKey(destinationKey)
        .build();

    CopyRequest copyRequest = CopyRequest.builder()
        .copyObjectRequest(copyObjectRequest)
        .build();

    Copy copy = transferManager.copy(copyRequest);

    CompletedCopy completedCopy = copy.completionFuture().join();
    return completedCopy.response().copyObjectResult().eTag();
}
```



**Note**

Um eine regionsübergreifende Kopie mit dem S3 Transfer Manager durchzuführen, aktivieren Sie `crossRegionAccessEnabled` auf dem AWS CRT-basierten S3-Client-Builder, wie im folgenden Codeausschnitt gezeigt.

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder()
    .crossRegionAccessEnabled(true)
    .build();

S3TransferManager transferManager = S3TransferManager.builder()
    .s3Client(s3AsyncClient)
    .build();
```

**Importe**

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;
```

**Hochladen eines lokalen Verzeichnisses in einen S3-Bucket**

Das folgende Beispiel zeigt, wie Sie ein lokales Verzeichnis in S3 hochladen können.

Rufen Sie zunächst die Methode [uploadDirectory](#) der `S3TransferManager` Instance auf und übergeben Sie ein [UploadDirectoryRequest](#).

Das [DirectoryUpload](#) Objekt stellt den Upload-Prozess dar, der ein generiert, [CompletedDirectoryUpload](#) wenn die Anforderung abgeschlossen ist. Das `CompletedDirectoryUpload` Objekt enthält Informationen über die Ergebnisse der Übertragung, einschließlich der Dateien, die nicht übertragen werden konnten.

```
public Integer uploadDirectory(S3TransferManager transferManager,
```

```
        URI sourceDirectory, String bucketName) {
            DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
                .source(Paths.get(sourceDirectory))
                .bucket(bucketName)
                .build());

            CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
            completedDirectoryUpload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
            return completedDirectoryUpload.failedTransfers().size();
        }
    }
```

## Importe

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;
```

## Herunterladen von S3-Bucket-Objekten in ein lokales Verzeichnis

Sie können die Objekte in einem S3-Bucket in ein lokales Verzeichnis herunterladen, wie im folgenden Beispiel gezeigt.

Um die Objekte in einem S3-Bucket in ein lokales Verzeichnis herunterzuladen, rufen Sie zunächst die [downloadDirectory](#)-Methode des Transfer Managers auf und übergeben eine [DownloadDirectoryRequest](#).

Das [-DirectoryDownload](#) Objekt stellt den Download-Prozess dar, der ein generiert, [CompletedDirectoryDownload](#) wenn die Anforderung abgeschlossen ist. Das

`CompletedDirectoryDownload` Objekt enthält Informationen über die Ergebnisse der Übertragung, einschließlich der Dateien, die nicht übertragen werden konnten.

```
public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
    URI destinationPathURI, String bucketName) {
    DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
    .destination(Paths.get(destinationPathURI))
    .bucket(bucketName)
    .build());
    CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryDownload.failedTransfers().size();
}
```

## Importe

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;

import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;
```

## Vollständige Beispiele anzeigen

[GitHub](#) enthält den vollständigen Code für alle Beispiele auf dieser Seite.

## Arbeiten mit Amazon Simple Notification Service

Mit Amazon Simple Notification Service können Sie ganz einfach Echtzeitbenachrichtigungen von Ihren Anwendungen über mehrere Kommunikationskanäle an Abonnenten senden. In diesem Thema wird beschrieben, wie Sie einige der Grundfunktionen von ausführenAmazon SNS.

### Erstellen eines Themas

Ein Thema ist eine logische Gruppierung von Kommunikationskanälen, die definiert, an welche Systeme eine Nachricht gesendet werden soll, z. B. das Ausfächern einer Nachricht an AWS Lambda und einen HTTP-Webhook. Sie senden Nachrichten an Amazon SNS und diese werden dann an die Kanäle verteilt, die im Thema definiert sind. Dadurch werden die Nachrichten für Abonnenten zur Verfügung gestellt.

Um ein Thema zu erstellen, erstellen Sie zunächst mithilfe der `name()` Methode im Builder ein [CreateTopicRequest](#) Objekt mit dem Namen des Themas. Senden Sie dann das Anforderungsobjekt an Amazon SNS, indem Sie die Methode `createTopic()` von [SnsClient](#) verwenden. Sie können das Ergebnis dieser Anfrage als [CreateTopicResponse](#) Objekt erfassen, wie im folgenden Codeausschnitt gezeigt.

#### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

#### Code

```
public static String createSNSTopic(SnsClient snsClient, String topicName ) {

    CreateTopicResponse result = null;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
```

```
        .name(topicName)
        .build();

        result = snsClient.createTopic(request);
        return result.topicArn();
    } catch (SnsException e) {

        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Listen Sie Ihre Amazon SNS Themen auf

Um eine Liste Ihrer vorhandenen Amazon SNS Themen abzurufen, erstellen Sie ein [ListTopicsRequest](#) Objekt. Senden Sie dann das Anforderungsobjekt an Amazon SNS, indem Sie die Methode `listTopics()` von `SnsClient` verwenden. Sie können das Ergebnis dieser Anfrage als [ListTopicsResponse](#) Objekt erfassen.

Der folgende Codeausschnitt druckt den HTTP-Statuscode der Anfrage und eine Liste der Amazon Resource Names (ARNs) für Ihre Amazon SNS-Themen aus.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

### Code

```
public static void listSNSTopics(SnsClient snsClient) {

    try {
        ListTopicsRequest request = ListTopicsRequest.builder()
            .build();

        ListTopicsResponse result = snsClient.listTopics(request);
    }
}
```

```
        System.out.println("Status was " + result.sdkHttpResponse().statusCode() +
"\n\nTopics\n\n" + result.topics());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Abonnieren eines Endpunkts für ein Thema

Nachdem Sie ein Thema erstellt haben, können Sie konfigurieren, welche Kommunikationskanäle Endpunkte für dieses Thema sein sollen. Nachrichten werden an diese Endpunkte verteilt, nachdem Amazon SNS diese empfangen hat.

Wenn Sie einen Kommunikationskanal als Endpunkt für ein Thema konfigurieren möchten, abonnieren Sie diesen Endpunkt für das Thema. Erstellen Sie zunächst ein [SubscribeRequest](#) Objekt. Geben Sie den Kommunikationskanal (z. B. lambda oder email) als `anprotocol()`. Stellen Sie das `endpoint()` auf den entsprechenden Ausgabespeicherort ein (z. B. den ARN einer Lambda Funktion oder eine E-Mail-Adresse), und legen Sie dann den ARN des Themas fest, das Sie abonnieren möchten, als `topicArn()`. Senden Sie das Anforderungsobjekt mit der `subscribe()` Methode von `SnsClient`. Amazon SNS Sie können das Ergebnis dieser Anfrage als [SubscribeResponse](#) Objekt erfassen.

Im folgenden Codeausschnitt wird dargestellt, wie Sie eine E-Mail-Adresse für ein Thema abonnieren.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
```

### Code

```
public static void subEmail(SnsClient snsClient, String topicArn, String email) {

    try {
```

```
SubscribeRequest request = SubscribeRequest.builder()
    .protocol("email")
    .endpoint(email)
    .returnSubscriptionArn(true)
    .topicArn(topicArn)
    .build();

SubscribeResponse result = snsClient.subscribe(request);
System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n\n"
    + "Status is " + result.sdkHttpResponse().statusCode());

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Veröffentlichen einer Nachricht für ein Thema

Nachdem Sie ein Thema und einen oder mehrere Endpunkte dafür konfiguriert haben, können Sie eine Nachricht dafür veröffentlichen. Erstellen Sie zunächst ein [PublishRequest](#) Objekt. Geben Sie zum Senden `message()` und den ARN des Themas (`topicArn()`) an, an das es gesendet werden soll. Senden Sie dann das Anforderungsobjekt an Amazon SNS, indem Sie die Methode `publish()` von `SnsClient` verwenden. Sie können das Ergebnis dieser Anfrage als [PublishResponse](#) Objekt erfassen.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

### Code

```
public static void pubTopic(SnsClient snsClient, String message, String topicArn) {

    try {
        PublishRequest request = PublishRequest.builder()
```

```
        .message(message)
        .topicArn(topicArn)
        .build();

    PublishResponse result = snsClient.publish(request);
    System.out.println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Abmelden eines Endpunkts von einem Thema

Sie können die als Endpunkte für ein Thema konfigurierten Kommunikationskanäle entfernen. Danach ist das Thema selbst weiterhin vorhanden und verteilt Nachrichten an alle anderen Endpunkte, die für dieses Thema konfiguriert sind.

Wenn Sie einen Kommunikationskanal als Endpunkt für ein Thema entfernen möchten, melden Sie diesen Endpunkt für das Thema ab. Erstellen Sie zunächst ein [UnsubscribeRequest](#) Objekt und legen Sie den ARN des Themas, von dem Sie sich abmelden möchten, als `subscriptionArn()`. Senden Sie dann das Anforderungsobjekt an SNS, indem Sie die Methode `unsubscribe()` von `SnsClient` verwenden. Sie können das Ergebnis dieser Anfrage als [UnsubscribeResponse](#) Objekt erfassen.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
```

### Code

```
public static void unSub(SnsClient snsClient, String subscriptionArn) {

    try {
```



```
UnsubscribeRequest request = UnsubscribeRequest.builder()
    .subscriptionArn(subscriptionArn)
    .build();

UnsubscribeResponse result = snsClient.unsubscribe(request);

System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode()
    + "\n\nSubscription was removed for " + request.subscriptionArn());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Löschen eines Themas

Um ein Amazon SNS Thema zu löschen, erstellen Sie zunächst ein [DeleteTopicRequest](#) Objekt mit dem ARN des Themas, der als `topicArn()` Methode im Builder festgelegt ist. Senden Sie dann das Anforderungsobjekt an Amazon SNS, indem Sie die Methode `deleteTopic()` von `SnsClient` verwenden. Sie können das Ergebnis dieser Anfrage als [DeleteTopicResponse](#) Objekt erfassen, wie im folgenden Codeausschnitt gezeigt.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

### Code

```
public static void deleteSNSTopic(SnsClient snsClient, String topicArn ) {

    try {
        DeleteTopicRequest request = DeleteTopicRequest.builder()
            .topicArn(topicArn)
            .build();
```

```
        DeleteTopicResponse result = snsClient.deleteTopic(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

Weitere Informationen finden Sie im [Amazon Simple Notification Service-Entwicklerhandbuch](#).

## Arbeiten mit Amazon Simple Queue Service

Dieser Abschnitt enthält Beispiele für die Programmierung [Amazon Simple Queue Service](#) mit AWS SDK for Java 2.x.

Die folgenden Beispiele enthalten nur den Code, der zur Demonstration jeder Technik nötig ist. Der [vollständige Beispielcode ist auf verfügbar GitHub](#). Von dort aus können Sie eine einzelne Quelldatei herunterladen oder das Repository klonen, um alle Beispiele lokal zu erstellen und auszuführen.

Themen

- [Arbeiten mit Amazon Simple Queue Service Nachrichtenwarteschlangen](#)
- [Senden, Empfangen und Löschen von Amazon Simple Queue Service Nachrichten](#)

## Arbeiten mit Amazon Simple Queue Service Nachrichtenwarteschlangen

Eine Nachrichtenwarteschlange ist der logische Container, der zum zuverlässigen Senden von Nachrichten in Amazon Simple Queue Service genutzt wird. Es gibt zwei Arten von Warteschlangen: Standard und First-in-First-out-Verfahren (FIFO). Weitere Informationen zu Warteschlangen und den Unterschieden zwischen diesen Typen finden Sie im [Amazon Simple Queue Service - Entwicklerhandbuch](#).

In diesem Thema wird beschrieben, wie Sie Amazon Simple Queue Service-Warteschlangen erstellen, auflisten, löschen und die URL einer Warteschlange mit AWS SDK for Java abrufen können.

Die `sqsClient` Variable, die in den folgenden Beispielen verwendet wird, kann aus dem folgenden Codeausschnitt erstellt werden.

```
SqsClient sqsClient = SqsClient.create();
```

Wenn Sie ein `SqsClient` mit der statischen `create()` Methode erstellen, konfiguriert das SDK die Region mithilfe der [standardmäßigen Regionsanbieterkette](#) und die Anmeldeinformationen mithilfe der [standardmäßigen Anbieterkette für Anmeldeinformationen](#).

## Erstellen einer Warteschlange

Verwenden Sie die `-SqsClient`'s `createQueue` Methode und stellen Sie ein [CreateQueueRequest](#) Objekt bereit, das die Warteschlangenparameter beschreibt, wie im folgenden Codeausschnitt gezeigt.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

### Code

```
CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
    .queueName(queueName)
    .build();

sqsClient.createQueue(createQueueRequest);
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Auflisten von Warteschlangen

Um die Amazon Simple Queue Service Warteschlangen für Ihr Konto aufzulisten, rufen Sie die `-SqsClient`'s `listQueues` Methode mit einem [ListQueuesRequest](#) Objekt auf.

Wenn Sie das Format der [listQueues](#) Methode verwenden, die keine Parameter verwendet, gibt der Service alle Warteschlangen zurück – bis zu 1 000 Warteschlangen.

Sie können dem [ListQueuesRequest](#) Objekt ein Präfix für den Warteschlangennamen bereitstellen, um die Ergebnisse auf Warteschlangen zu beschränken, die diesem Präfix entsprechen, wie im folgenden Code gezeigt.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

### Code

```
String prefix = "que";

try {
    ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
    ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);

    for (String url : listQueuesResponse.queueUrls()) {
        System.out.println(url);
    }

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Abrufen der URL für eine Warteschlange

Der folgende Code zeigt, wie Sie die URL für eine Warteschlange abrufen, indem Sie die `-SqsClient` 's `getQueueUrl` Methode mit einem [GetQueueUrlRequest](#) Objekt aufrufen.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
```

```
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

## Code

```
GetQueueUrlResponse getQueueUrlResponse =

sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
String queueUrl = getQueueUrlResponse.queueUrl();
return queueUrl;
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Löschen einer Warteschlange

Geben Sie die [URL](#) der Warteschlange für das [DeleteQueueRequest](#) Objekt an. Rufen Sie dann die `-SqsClient`'s `deleteQueue` Methode auf, um eine Warteschlange zu löschen, wie im folgenden Code gezeigt.

## Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

## Code

```
public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {

    try {

        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();

        DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
            .queueUrl(queueUrl)
            .build();
```

```
        sqsClient.deleteQueue(deleteQueueRequest);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Weitere Informationen

- [CreateQueue](#) in der Amazon Simple Queue Service API-Referenz zu
- [GetQueueUrl](#) in der Amazon Simple Queue Service API-Referenz zu
- [ListQueues](#) in der Amazon Simple Queue Service API-Referenz zu
- [DeleteQueue](#) in der Amazon Simple Queue Service API-Referenz zu

## Senden, Empfangen und Löschen von Amazon Simple Queue Service Nachrichten

Eine Nachricht ist ein Datenabschnitt, der von verteilten Komponenten gesendet und empfangen werden kann. Nachrichten werden immer mit einer [SQS-Warteschlange](#) geliefert.

Die `sqsClient` Variable, die in den folgenden Beispielen verwendet wird, kann aus dem folgenden Codeausschnitt erstellt werden.

```
SqsClient sqsClient = SqsClient.create();
```

Wenn Sie ein `SqsClient` mit der statischen `create()` Methode erstellen, konfiguriert das SDK die Region mithilfe der [standardmäßigen Regionsanbieterkette](#) und die Anmeldeinformationen mithilfe der [standardmäßigen Anbieterkette für Anmeldeinformationen](#).

## Senden einer Nachricht

Fügen Sie einer -Amazon Simple Queue Service Warteschlange eine einzelne Nachricht hinzu, indem Sie die `SqsClient Client-sendMessage` Methode aufrufen. Geben Sie ein

[SendMessageRequest](#) Objekt an, das die [URL](#) der Warteschlange, den Nachrichtentext und den optionalen Verzögerungswert (in Sekunden) enthält.

## Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

## Code

```
sqsClient.sendMessage(SendMessageRequest.builder()
    .queueUrl(queueUrl)
    .messageBody("Hello world!")
    .delaySeconds(10)
    .build());

sqsClient.sendMessage(sendMsgRequest);
```

## Senden mehrerer Nachrichten in einer Anforderung

Senden mehrerer Nachrichten in einer einzigen Anforderung unter Verwendung der `sendMessageBatch`-Methode des `SqsClient`. Diese Methode verwendet eine [SendMessageBatchRequest](#), die die Warteschlangen-URL und eine Liste der zu sendenden Nachrichten enthält. (Jede Nachricht ist ein [SendMessageBatchRequestEntry](#).) Sie können das Senden einer bestimmten Nachricht auch verzögern, indem Sie einen Verzögerungswert für die Nachricht festlegen.

## Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

## Code

```
SendMessageBatchRequest sendMessageBatchRequest =
    SendMessageBatchRequest.builder()
```

```
        .queueUrl(queueUrl)

        .entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from msg
1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10).build())
        .build();
    sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Nachrichten abrufen

Abrufen von aktuell in der Warteschlange enthaltenen Nachrichten durch Aufruf der `receiveMessage`-Methode des `SqsClient`. Diese Methode verwendet eine [ReceiveMessageRequest](#), die die Warteschlangen-URL enthält. Sie können auch die maximale Anzahl von Nachrichten angeben, die zurückgegeben werden soll. Nachrichten werden als Liste von [Message](#)-Objekten zurückgegeben.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

### Code

```
    try {
        ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
            .queueUrl(queueUrl)
            .numberOfMessages(5)
            .build();
        List<Message> messages =
sqsClient.receiveMessage(receiveMessageRequest).messages();
        return messages;
    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
```



```
return null;
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Löschen einer Nachricht nach Erhalt

Nachdem Sie eine Nachricht empfangen und ihren Inhalt verarbeitet haben, löschen Sie die Nachricht aus der Warteschlange, indem Sie den Empfangs-Handle und die Warteschlangen-URL der Nachricht an die `SqsClient`'s [deleteMessage](#) Methode senden.

### Importe

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

### Code

```
try {
    for (Message message : messages) {
        DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .build();
        sqsClient.deleteMessage(deleteMessageRequest);
    }
}
```

Sehen Sie sich das [vollständige Beispiel](#) auf an GitHub.

## Weitere Infos

- [Funktionsweise von Amazon Simple Queue ServiceWarteschlangen](#) im -Amazon Simple Queue ServiceEntwicklerhandbuch
- [SendMessage](#) in der Amazon Simple Queue Service API-Referenz zu
- [SendMessageBatch](#) in der Amazon Simple Queue Service API-Referenz zu
- [ReceiveMessage](#) in der Amazon Simple Queue Service API-Referenz zu
- [DeleteMessage](#) in der Amazon Simple Queue Service API-Referenz zu

## Arbeiten mit Amazon Transcribe

Das folgende Beispiel zeigt, wie bidirektionales Streaming mit Amazon Transcribe funktioniert. Bidirektionales Streaming bedeutet, dass ein Datenstrom zum Service gesendet wird und auch in Echtzeit wieder empfangen wird. Das Beispiel verwendet Amazon Transcribe-Streaming-Transkription zum Senden eines Audio-Streams und Empfangen eines transkribierten Text-Streams in Echtzeit.

Weitere Informationen zu dieser Funktion finden Sie unter [Streaming-Transkription](#) im Amazon Transcribe Entwicklerhandbuch.

Informationen [zu den ersten](#) Schritten finden Sie unter Amazon Transcribe Erste Schritte im Amazon Transcribe Entwicklerhandbuch.

## Richten Sie das Mikrofon ein

Dieser Code verwendet das `javax.sound.sampled`-Paket zum Streamen von Audio aus einem Eingabegerät.

### Code

```
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.TargetDataLine;

public class Microphone {

    public static TargetDataLine get() throws Exception {
        AudioFormat format = new AudioFormat(16000, 16, 1, true, false);
        DataLine.Info datalineInfo = new DataLine.Info(TargetDataLine.class, format);

        TargetDataLine dataLine = (TargetDataLine) AudioSystem.getLine(datalineInfo);
        dataLine.open(format);

        return dataLine;
    }
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Erstellen Sie einen Herausgeber

Dieser Code implementiert einen Herausgeber, der Audiodaten aus dem Amazon Transcribe-Audio-Stream veröffentlicht.

### Code

```
package com.amazonaws.transcribe;

import java.io.IOException;
import java.io.InputStream;
import java.io.UncheckedIOException;
import java.nio.ByteBuffer;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.atomic.AtomicLong;
import org.reactivestreams.Publisher;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.transcribestreaming.model.AudioEvent;
import software.amazon.awssdk.services.transcribestreaming.model.AudioStream;
import
    software.amazon.awssdk.services.transcribestreaming.model.TranscribeStreamingException;

public class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;

    public AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {
        s.onSubscribe(new SubscriptionImpl(s, inputStream));
    }

    private class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);
    }
}
```

```
private final Subscriber<? super AudioStream> subscriber;
private final InputStream inputStream;

private SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
inputStream) {
    this.subscriber = s;
    this.inputStream = inputStream;
}

@Override
public void request(long n) {
    if (n <= 0) {
        subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
    }

    demand.getAndAdd(n);

    executor.submit(() -> {
        try {
            do {
                ByteBuffer audioBuffer = getNextEvent();
                if (audioBuffer.remaining() > 0) {
                    AudioEvent audioEvent = audioEventFromBuffer(audioBuffer);
                    subscriber.onNext(audioEvent);
                } else {
                    subscriber.onComplete();
                    break;
                }
            } while (demand.decrementAndGet() > 0);
        } catch (TranscribeStreamingException e) {
            subscriber.onError(e);
        }
    });
}

@Override
public void cancel() {
}

private ByteBuffer getNextEvent() {
    ByteBuffer audioBuffer;
    byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];
```

```
int len = 0;
try {
    len = inputStream.read(audioBytes);

    if (len <= 0) {
        audioBuffer = ByteBuffer.allocate(0);
    } else {
        audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
    }
} catch (IOException e) {
    throw new UncheckedIOException(e);
}

return audioBuffer;
}

private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
    return AudioEvent.builder()
        .audioChunk(SdkBytes.fromByteBuffer(bb))
        .build();
}
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Erstellen Sie den Client und starten Sie den Stream

Erstellen Sie in der Hauptmethode ein Anforderungsobjekt, starten Sie den Audioeingabe-Stream und instanziiieren Sie den Herausgeber mit der Audioeingabe.

Sie müssen auch eine erstellen [StartStreamTranscriptionResponseHandler](#), um anzugeben, wie mit der Antwort von umgegangen werden soll Amazon Transcribe.

Verwenden Sie dann die `startStreamTranscription` Methode `TranscribeStreamingAsyncClient`'s, um das bidirektionale Streaming zu starten.

### Importe

```
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
```

```
import javax.sound.sampled.TargetDataLine;
import javax.sound.sampled.AudioInputStream;
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.transcribestreaming.TranscribeStreamingAsyncClient;
import
    software.amazon.awssdk.services.transcribestreaming.model.TranscribeStreamingException ;
import
    software.amazon.awssdk.services.transcribestreaming.model.StartStreamTranscriptionRequest;
import software.amazon.awssdk.services.transcribestreaming.model.MediaEncoding;
import software.amazon.awssdk.services.transcribestreaming.model.LanguageCode;
import
    software.amazon.awssdk.services.transcribestreaming.model.StartStreamTranscriptionResponseHandler;
import software.amazon.awssdk.services.transcribestreaming.model.TranscriptEvent;
```

## Code

```
public static void convertAudio(TranscribeStreamingAsyncClient client) throws
Exception {

    try {

        StartStreamTranscriptionRequest request =
StartStreamTranscriptionRequest.builder()
            .mediaEncoding(MediaEncoding.PCM)
            .languageCode(LanguageCode.EN_US)
            .mediaSampleRateHertz(16_000).build();

        TargetDataLine mic = Microphone.get();
        mic.start();

        AudioStreamPublisher publisher = new AudioStreamPublisher(new
AudioInputStream(mic));

        StartStreamTranscriptionResponseHandler response =
            StartStreamTranscriptionResponseHandler.builder().subscriber(e -> {
                TranscriptEvent event = (TranscriptEvent) e;
                event.transcript().results().forEach(r ->
                    r.alternatives().forEach(a -> System.out.println(a.transcript())));
            }).build();

        // Keeps Streaming until you end the Java program
        client.startStreamTranscription(request, publisher, response);
```

```
    } catch (TranscribeStreamingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

Das [vollständige Beispiel](#) finden Sie unter. GitHub

## Weitere Informationen

- [Wie es funktioniert](#), finden Sie im Amazon Transcribe Entwicklerhandbuch.
- [Erste Schritte mit dem Streamen von Audio](#) im Amazon Transcribe Entwicklerhandbuch.

# Codebeispiele für SDK für Java 2.x

Die Codebeispiele in diesem Thema zeigen Ihnen, wie Sie mit verwenden AWS SDK for Java 2.x AWS.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Serviceübergreifende Beispiele sind Beispielanwendungen, die über mehrere AWS-Services hinweg arbeiten.

## Beispiele

- [Aktionen und Szenarien mit SDK for Java 2.x](#)
- [Serviceübergreifende Beispiele mit SDK for Java 2.x](#)

## Aktionen und Szenarien mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit verwenden AWS-Services.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

## Services

- [API Gateway-Beispiele mit SDK für Java 2.x](#)
- [Beispiele für Application Auto Scaling mit SDK for Java 2.x](#)
- [Beispiele für Application Recovery Controller unter Verwendung des SDK für Java 2.x](#)
- [Aurora-Beispiele mit SDK for Java 2.x](#)



- [Auto Scaling-Beispiele mit SDK für Java 2.x](#)
- [Amazon-Bedrock-Beispiele mit SDK for Java 2.x](#)
- [Amazon-Bedrock-Laufzeitbeispiele mit SDK für Java 2.x](#)
- [CloudFront -Beispiele mit SDK for Java 2.x](#)
- [CloudWatch -Beispiele mit SDK for Java 2.x](#)
- [CloudWatch Beispiele für Ereignisse mit SDK for Java 2.x](#)
- [CloudWatch Protokolliert Beispiele mit SDK for Java 2.x](#)
- [Beispiele für Amazon Cognito Identity unter Verwendung von SDK for Java 2.x](#)
- [Beispiele für Amazon Cognito-Identitätsanbieter mit SDK for Java 2.x](#)
- [Amazon Comprehend-Beispiele mit SDK for Java 2.x](#)
- [DynamoDB-Beispiele mit SDK for Java 2.x](#)
- [Amazon EC2-Beispiele mit SDK for Java 2.x](#)
- [Amazon-ECS-Beispiele mit SDK for Java 2.x](#)
- [Beispiele für Elastic Load Balancing mit SDK for Java 2.x](#)
- [MediaStore -Beispiele mit SDK for Java 2.x](#)
- [OpenSearch Servicebeispiele mit SDK for Java 2.x](#)
- [EventBridge -Beispiele mit SDK for Java 2.x](#)
- [Prognosebeispiele mit SDK for Java 2.x](#)
- [AWS Glue -Beispiele mit SDK for Java 2.x](#)
- [HealthImaging -Beispiele mit SDK for Java 2.x](#)
- [IAM-Beispiele mit SDK for Java 2.x](#)
- [AWS IoT -Beispiele mit SDK for Java 2.x](#)
- [AWS IoT data -Beispiele mit SDK for Java 2.x](#)
- [Beispiele für Amazon Keyspaces mit SDK for Java 2.x](#)
- [Kinesis-Beispiele mit SDK for Java 2.x](#)
- [AWS KMS -Beispiele mit SDK for Java 2.x](#)
- [Lambda-Beispiele mit SDK für Java 2.x](#)
- [MediaConvert -Beispiele mit SDK for Java 2.x](#)
- [Migration Hub-Beispiele mit SDK for Java 2.x](#)

- [Amazon-Personalize-Beispiele mit SDK für Java 2.x](#)
- [Beispiele für Amazon Personalize Events mit SDK for Java 2.x](#)
- [Beispiele für Amazon Personalize Runtime mit SDK for Java 2.x](#)
- [Amazon Pinpoint-Beispiele mit SDK for Java 2.x](#)
- [Amazon Pinpoint-SMS- und Sprachnachrichten-API-Beispiele mit SDK for Java 2.x](#)
- [Amazon Polly-Beispiele mit SDK for Java 2.x](#)
- [Amazon-RDS-Beispiele mit SDK für Java 2.x](#)
- [Amazon-Redshift-Beispiele mit SDK für Java 2.x](#)
- [Amazon Rekognition-Beispiele mit SDK for Java 2.x](#)
- [Beispiele für die Route-53-Domainregistrierung mit SDK for Java 2.x](#)
- [Amazon S3-Beispiele mit SDK for Java 2.x](#)
- [S3-Glacier-Beispiele mit SDK für Java 2.x](#)
- [SageMaker -Beispiele mit SDK for Java 2.x](#)
- [Secrets-Manager-Beispiele mit SDK für Java 2.x](#)
- [Amazon SES-Beispiele mit SDK for Java 2.x](#)
- [Amazon SES-API-v2-Beispiele mit SDK für Java 2.x](#)
- [Amazon SNS-Beispiele mit SDK for Java 2.x](#)
- [Amazon SQS-Beispiele mit SDK for Java 2.x](#)
- [Step Functions-Beispiele mit SDK für Java 2.x](#)
- [AWS STS -Beispiele mit SDK for Java 2.x](#)
- [AWS Support -Beispiele mit SDK for Java 2.x](#)
- [Systems Manager-Beispiele mit SDK für Java 2.x](#)
- [Amazon-Textract-Beispiele unter Verwendung von SDK for Java 2.x](#)
- [Amazon TranscribeBeispiele mit SDK for Java 2.x](#)

## API Gateway-Beispiele mit SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit API Gateway verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)

## Aktionen

### Erstellen einer REST-API

Das folgende Codebeispiel zeigt, wie Sie eine API Gateway-REST-API erstellen.

SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createAPI(ApiGatewayClient apiGateway, String restApiId,
String restApiName) {

    try {
        CreateRestApiRequest request = CreateRestApiRequest.builder()
            .cloneFrom(restApiId)
            .description("Created using the Gateway Java API")
            .name(restApiName)
            .build();

        CreateRestApiResponse response = apiGateway.createRestApi(request);
        System.out.println("The id of the new api is " + response.id());
        return response.id();
    }
}
```

```
    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Weitere API-Informationen finden Sie unter [CreateRestApi](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen einer REST-API

Das folgende Codebeispiel zeigt, wie Sie eine API Gateway-REST-API löschen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void deleteAPI(ApiGatewayClient apiGateway, String restApiId) {

    try {
        DeleteRestApiRequest request = DeleteRestApiRequest.builder()
            .restApiId(restApiId)
            .build();

        apiGateway.deleteRestApi(request);
        System.out.println("The API was successfully deleted");

    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteRestApi](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen einer Bereitstellung

Das folgende Codebeispiel zeigt, wie Sie eine Bereitstellung löschen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void deleteSpecificDeployment(ApiGatewayClient apiGateway, String
restApiId, String deploymentId) {

    try {
        DeleteDeploymentRequest request = DeleteDeploymentRequest.builder()
            .restApiId(restApiId)
            .deploymentId(deploymentId)
            .build();

        apiGateway.deleteDeployment(request);
        System.out.println("Deployment was deleted");

    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteDeployment](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Bereitstellen einer REST-API

Das folgende Codebeispiel zeigt, wie Sie eine API Gateway-REST-API bereitstellen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createNewDeployment(ApiGatewayClient apiGateway, String
restApiId, String stageName) {

    try {
        CreateDeploymentRequest request = CreateDeploymentRequest.builder()
            .restApiId(restApiId)
            .description("Created using the AWS API Gateway Java API")
            .stageName(stageName)
            .build();

        CreateDeploymentResponse response =
apiGateway.createDeployment(request);
        System.out.println("The id of the deployment is " + response.id());
        return response.id();

    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Weitere API-Informationen finden Sie unter [CreateDeployment](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Beispiele für Application Auto Scaling mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Application Auto Scaling verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)

## Aktionen

Deaktiviert eine Ressource

Das folgende Codebeispiel zeigt, wie Sie eine Application Auto Scaling-Ressource deaktivieren.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingExceptio
import
    software.amazon.awssdk.services.applicationautoscaling.model.DeleteScalingPolicyRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DeregisterScalableTargetReques
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRespons
```

```
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResponse;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DisableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableId> <policyName>\s

            Where:
                tableId - The table Id value (for example, table/Music).\s
                policyName - The name of the policy (for example, $Music5-scaling-
policy).

            """;
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        ApplicationAutoScalingClient appAutoScalingClient =
ApplicationAutoScalingClient.builder()
    .region(Region.US_EAST_1)
    .build();

        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
```



```
String tableId = args[0];
String policyName = args[1];

deletePolicy(appAutoScalingClient, policyName, tableWCUs, ns, tableId);
verifyScalingPolicies(appAutoScalingClient, tableId, ns, tableWCUs);
deregisterScalableTarget(appAutoScalingClient, tableId, ns, tableWCUs);
verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
}

public static void deletePolicy(ApplicationAutoScalingClient
appAutoScalingClient, String policyName, ScalableDimension tableWCUs,
ServiceNamespace ns, String tableId) {
    try {
        DeleteScalingPolicyRequest delSPRequest =
DeleteScalingPolicyRequest.builder()
        .policyName(policyName)
        .scalableDimension(tableWCUs)
        .serviceNamespace(ns)
        .resourceId(tableId)
        .build();

        appAutoScalingClient.deleteScalingPolicy(delSPRequest);
        System.out.println(policyName + " was deleted successfully.");

    } catch (ApplicationAutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

// Verify that the scaling policy was deleted
public static void verifyScalingPolicies(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
    DescribeScalingPoliciesRequest dscRequest =
DescribeScalingPoliciesRequest.builder()
    .scalableDimension(tableWCUs)
    .serviceNamespace(ns)
    .resourceId(tableId)
    .build();

    DescribeScalingPoliciesResponse response =
appAutoScalingClient.describeScalingPolicies(dscRequest);
    System.out.println("DescribeScalableTargets result: ");
    System.out.println(response);
}
```

```
}

    public static void deregisterScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
    try {
        DeregisterScalableTargetRequest targetRequest =
DeregisterScalableTargetRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceId(tableId)
            .build();

        appAutoScalingClient.deregisterScalableTarget(targetRequest);
        System.out.println("The scalable target was deregistered.");

    } catch (ApplicationAutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

    public static void verifyTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
        DescribeScalableTargetsRequest dscRequest =
DescribeScalableTargetsRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceIds(tableId)
            .build();

        DescribeScalableTargetsResponse response =
appAutoScalingClient.describeScalableTargets(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteScalingPolicy](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Registriert eine Ressource

Das folgende Codebeispiel zeigt, wie Sie eine Application Auto Scaling-Ressource registrieren.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingException;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsResponse;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResponse;
import software.amazon.awssdk.services.applicationautoscaling.model.PolicyType;
import
    software.amazon.awssdk.services.applicationautoscaling.model.PredefinedMetricSpecification;
import
    software.amazon.awssdk.services.applicationautoscaling.model.PutScalingPolicyRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.RegisterScalableTargetRequest;
import software.amazon.awssdk.services.applicationautoscaling.model.ScalingPolicy;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import software.amazon.awssdk.services.applicationautoscaling.model.MetricType;
import
    software.amazon.awssdk.services.applicationautoscaling.model.TargetTrackingScalingPolicyConfiguration;
import java.util.List;

/**
```

```
* Before running this Java V2 code example, set up your development environment,
including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class EnableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableId> <roleARN> <policyName>\s

            Where:
                tableId - The table Id value (for example, table/Music).
                roleARN - The ARN of the role that has ApplicationAutoScaling
permissions.
                policyName - The name of the policy to create.

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        System.out.println("This example registers an Amazon DynamoDB table, which
is the resource to scale.");
        String tableId = args[0];
        String roleARN = args[1];
        String policyName = args[2];
        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        ApplicationAutoScalingClient appAutoScalingClient =
ApplicationAutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        registerScalableTarget(appAutoScalingClient, tableId, roleARN, ns,
tableWCUs);
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
    }
}
```

```
        configureScalingPolicy(appAutoScalingClient, tableId, ns, tableWCUs,
policyName);
    }

    public static void registerScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, String roleARN, ServiceNamespace ns,
ScalableDimension tableWCUs) {
        try {
            RegisterScalableTargetRequest targetRequest =
RegisterScalableTargetRequest.builder()
                .serviceNamespace(ns)
                .scalableDimension(tableWCUs)
                .resourceId(tableId)
                .roleARN(roleARN)
                .minCapacity(5)
                .maxCapacity(10)
                .build();

            appAutoScalingClient.registerScalableTarget(targetRequest);
            System.out.println("You have registered " + tableId);

        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Verify that the target was created.
    public static void verifyTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
        DescribeScalableTargetsRequest dscRequest =
DescribeScalableTargetsRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceIds(tableId)
            .build();

        DescribeScalableTargetsResponse response =
appAutoScalingClient.describeScalableTargets(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }

    // Configure a scaling policy.
```

```
public static void configureScalingPolicy(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs, String policyName) {
    // Check if the policy exists before creating a new one.
    DescribeScalingPoliciesResponse describeScalingPoliciesResponse =
appAutoScalingClient.describeScalingPolicies(DescribeScalingPoliciesRequest.builder()
    .serviceNamespace(ns)
    .resourceId(tableId)
    .scalableDimension(tableWCUs)
    .build());

    if (!describeScalingPoliciesResponse.scalingPolicies().isEmpty()) {
        // If policies exist, consider updating an existing policy instead of
creating a new one.
        System.out.println("Policy already exists. Consider updating it
instead.");
        List<ScalingPolicy> polList =
describeScalingPoliciesResponse.scalingPolicies();
        for (ScalingPolicy pol : polList) {
            System.out.println("Policy name:" +pol.policyName());
        }
    } else {
        // If no policies exist, proceed with creating a new policy.
        PredefinedMetricSpecification specification =
PredefinedMetricSpecification.builder()

        .predefinedMetricType(MetricType.DYNAMO_DB_WRITE_CAPACITY_UTILIZATION)
        .build();

        TargetTrackingScalingPolicyConfiguration policyConfiguration =
TargetTrackingScalingPolicyConfiguration.builder()
        .predefinedMetricSpecification(specification)
        .targetValue(50.0)
        .scaleInCooldown(60)
        .scaleOutCooldown(60)
        .build();

        PutScalingPolicyRequest putScalingPolicyRequest =
PutScalingPolicyRequest.builder()
        .targetTrackingScalingPolicyConfiguration(policyConfiguration)
        .serviceNamespace(ns)
        .scalableDimension(tableWCUs)
        .resourceId(tableId)
        .policyName(policyName)
```

```
        .policyType(PolicyType.TARGET_TRACKING_SCALING)
        .build();

    try {
        appAutoScalingClient.putScalingPolicy(putScalingPolicyRequest);
        System.out.println("You have successfully created a scaling policy
for an Application Auto Scaling scalable target");
    } catch (ApplicationAutoScalingException e) {
        System.err.println("Error: " + e.awsErrorDetails().errorMessage());
    }
}
}
```

- Weitere API-Informationen finden Sie unter [RegisterScalableTarget](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Beispiele für Application Recovery Controller unter Verwendung des SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Application Recovery Controller verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Themen

- [Aktionen](#)

## Aktionen

### Abrufen des Status eines Routing-Steuerelements

Das folgende Codebeispiel zeigt, wie Sie den Status einer Application Recovery Controller-Routing-Steuerung abrufen.

SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static GetRoutingControlStateResponse
getRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
    String routingControlArn) {
    // As a best practice, we recommend choosing a random cluster endpoint to
    get or
    // set routing control states.
    // For more information, see
    // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
    practices.html#route53-arc-best-practices.regional
    Collections.shuffle(clusterEndpoints);
    for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
        try {
            System.out.println(clusterEndpoint);
            Route53RecoveryClusterClient client =
Route53RecoveryClusterClient.builder()
                .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                .region(Region.of(clusterEndpoint.region())).build();
            return client.getRoutingControlState(
                GetRoutingControlStateRequest.builder()
                    .routingControlArn(routingControlArn).build());
        } catch (Exception exception) {
            System.out.println(exception);
        }
    }
    return null;
}
```



- Weitere API-Informationen finden Sie unter [GetRoutingControlState](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Aktualisieren des Status eines Routing-Steuerelements

Das folgende Codebeispiel zeigt, wie Sie den Status einer Application Recovery Controller-Routing-Steuerung aktualisieren.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

public static UpdateRoutingControlStateResponse
updateRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
    String routingControlArn,
    String routingControlState) {
    // As a best practice, we recommend choosing a random cluster endpoint to
    get or
    // set routing control states.
    // For more information, see
    // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
    practices.html#route53-arc-best-practices.regional
    Collections.shuffle(clusterEndpoints);
    for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
        try {
            System.out.println(clusterEndpoint);
            Route53RecoveryClusterClient client =
Route53RecoveryClusterClient.builder()
                .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                .region(Region.of(clusterEndpoint.region()))
                .build();
            return client.updateRoutingControlState(
                UpdateRoutingControlStateRequest.builder()

.routingControlArn(routingControlArn).routingControlState(routingControlState).build());
        } catch (Exception exception) {
            System.out.println(exception);
        }
    }
}

```

```
    }  
  }  
  return null;  
}
```

- Weitere API-Informationen finden Sie unter [UpdateRoutingControlState](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Aurora-Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Aurora verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Erste Schritte

#### Hello Aurora

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Aurora.

#### SDK für Java 2.x

##### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.paginators.DescribeDBClustersIterable;

public class DescribeDbClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        describeClusters(rdsClient);
        rdsClient.close();
    }

    public static void describeClusters(RdsClient rdsClient) {
        DescribeDBClustersIterable clustersIterable =
rdsClient.describeDBClustersPaginator();
        clustersIterable.stream()
            .flatMap(r -> r.dbClusters().stream())
            .forEach(cluster -> System.out
                .println("Database name: " + cluster.databaseName() + " Arn
= " + cluster.dbClusterArn()));
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeDBClusters](#) in der API-Referenz zu AWS SDK for Java 2.x .

## Themen

- [Aktionen](#)
- [Szenarien](#)

## Aktionen

### DB-Cluster erstellen

Das folgende Codebeispiel zeigt, wie Sie einen Aurora-DB-Cluster erstellen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
    String dbClusterIdentifier, String userName, String password) {
    try {
        CreateDbClusterRequest clusterRequest = CreateDbClusterRequest.builder()
            .databaseName(dbName)
            .dbClusterIdentifier(dbClusterIdentifier)
            .dbClusterParameterGroupName(dbParameterGroupFamily)
            .engine("aurora-mysql")
            .masterUsername(userName)
            .masterUserPassword(password)
            .build();

        CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
        return response.dbCluster().dbClusterArn();

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

- Weitere API-Informationen finden Sie unter [CreateDBCluster](#) in der API-Referenz zu AWS SDK for Java 2.x .

## Erstellen einer DB-Cluster-Parametergruppe

Das folgende Codebeispiel zeigt, wie Sie eine Aurora-DB-Cluster-Parametergruppe erstellen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
    String dbParameterGroupFamily) {
    try {
        CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .description("Created by using the AWS SDK for Java")
            .build();

        CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [CreateDBClusterParameterGroup](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erstellen eines DB-Cluster-Snapshots

Das folgende Codebeispiel zeigt, wie Sie einen Aurora-DB-Cluster-Snapshot erstellen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
    String dbSnapshotIdentifier) {
    try {
        CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
        System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [CreateDBClusterSnapshot](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erstellen einer DB-Instance in einem DB-Cluster

Das folgende Codebeispiel zeigt, wie Sie eine DB-Instance in einem Aurora-DB-Cluster erstellen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createDBInstanceCluster(RdsClient rdsClient,
      String dbInstanceIdentifier,
      String dbInstanceClusterIdentifier,
      String instanceClass) {
    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .engine("aurora-mysql")
            .dbInstanceClass(instanceClass)
            .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
        return response.dbInstance().dbInstanceArn();

    } catch (RdsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- Weitere API-Informationen finden Sie unter [CreateDBInstance](#) in der AWS SDK for Java 2.x - API-Referenz.

## Löschen eines DB-Clusters

Das folgende Codebeispiel zeigt, wie Sie einen Aurora-DB-Cluster löschen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
    try {
        DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .skipFinalSnapshot(true)
            .build();

        rdsClient.deleteDBCluster(deleteDbClusterRequest);
        System.out.println(dbInstanceClusterIdentifier + " was deleted!");


    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteDBCluster](#) in der API-Referenz zu AWS SDK for Java 2.x .

## Löschen einer DB-Cluster-Parametergruppe

Das folgende Codebeispiel zeigt, wie Sie eine Aurora-DB-Cluster-Parametergruppe löschen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.



```

    public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
        throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            didFind = false;
            int index = 1;
            for (DBInstance instance : instanceList) {
                instanceARN = instance.dbInstanceArn();
                if (instanceARN.compareTo(clusterDBARN) == 0) {
                    System.out.println(clusterDBARN + " still exists");
                    didFind = true;
                }
            }
            if ((index == listSize) && (!didFind)) {
                // Went through the entire list and did not find the
database ARN.

                isDataDel = true;
            }
            Thread.sleep(sleepTime * 1000);
            index++;
        }

        DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
            .builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .build();

        rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
        System.out.println(dbClusterGroupName + " was deleted.");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
    }
}

```

```
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteDBClusterParameterGroup](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Löschen einer DB-Instance

Das folgende Codebeispiel zeigt, wie Sie eine Aurora-DB-Instance löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .deleteAutomatedBackups(true)
        .skipFinalSnapshot(true)
        .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteDBInstance](#) in der API-Referenz zu AWS SDK for Java 2.x .

## Beschreiben von DB-Cluster-Parametergruppen

Das folgende Codebeispiel zeigt, wie Sie Aurora-DB-Cluster-Parametergruppen beschreiben.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
    try {
        DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .maxRecords(20)
            .build();

        List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
            .dbClusterParameterGroups();
        for (DBClusterParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbClusterParameterGroupName());
            System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeDBClusterParameterGroups](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Beschreiben von DB-Cluster-Snapshots

Das folgende Codebeispiel zeigt, wie Sie Aurora-DB-Cluster-Snapshots beschreiben.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
    String dbInstanceClusterIdentifier) {
    try {
        boolean snapshotReady = false;
        String snapshotReadyStr;
        System.out.println("Waiting for the snapshot to become available.");

        DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
            .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .build();

        while (!snapshotReady) {
            DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
            List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
            for (DBClusterSnapshot snapshot : snapshotList) {
                snapshotReadyStr = snapshot.status();
                if (snapshotReadyStr.contains("available")) {
                    snapshotReady = true;
                } else {
                    System.out.println(".");
                    Thread.sleep(sleepTime * 5000);
                }
            }
        }
    }
}
```

```
        }
    }

    System.out.println("The Snapshot is available!");

} catch (RdsException | InterruptedException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [DescribeDBClusterSnapshots](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Beschreiben von DB-Clustern

Das folgende Codebeispiel zeigt, wie Sie Aurora-DB-Cluster beschreiben.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void describeDbClusterParameters(RdsClient rdsClient, String
dbClusterGroupName, int flag) {
    try {
        DescribeDbClusterParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .build();
        } else {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .source("user")
```

```

        .build();
    }

    DescribeDbClusterParametersResponse response = rdsClient
        .describeDBClusterParameters(dbParameterGroupsRequest);
    List<Parameter> dbParameters = response.parameters();
    String paraName;
    for (Parameter para : dbParameters) {
        // Only print out information about either auto_increment_offset or
        // auto_increment_increment.
        paraName = para.parameterName();
        if ((paraName.compareTo("auto_increment_offset") == 0)
            || (paraName.compareTo("auto_increment_increment ") == 0)) {
            System.out.println("*** The parameter name is " + paraName);
            System.out.println("*** The parameter value is " +
para.parameterValue());
            System.out.println("*** The parameter data type is " +
para.dataType());
            System.out.println("*** The parameter description is " +
para.description());
            System.out.println("*** The parameter allowed values is " +
para.allowedValues());
        }
    }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}


```

- Weitere API-Informationen finden Sie unter [DescribeDBClusters](#) in der API-Referenz zu AWS SDK for Java 2.x .

## Beschreiben von DB-Instances

Das folgende Codebeispiel zeigt, wie Sie Aurora-DB-Instances beschreiben.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");
    try {
        DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
            .dbClusterIdentifier(dbClusterIdentifier)
            .build();

        while (!instanceReady) {
            DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
            List<DBCluster> clusterList = response.dbClusters();
            for (DBCluster cluster : clusterList) {
                instanceReadyStr = cluster.status();
                if (instanceReadyStr.contains("available")) {
                    instanceReady = true;
                } else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }
        System.out.println("Database cluster is available!");
    } catch (RdsException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeDBInstances](#) in der API-Referenz zu AWS SDK for Java 2.x .

## Beschreiben der Datenbank-Engine-Versionen

Das folgende Codebeispiel zeigt, wie Versionen der Aurora-Datenbank-Engine beschrieben werden.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .engine("aurora-mysql")
            .defaultOnly(true)
            .maxRecords(20)
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engineOb : engines) {
            System.out.println("The name of the DB parameter group family for
the database engine is "
                + engineOb.dbParameterGroupFamily());
            System.out.println("The name of the database engine " +
engineOb.engine());
            System.out.println("The version number of the database engine " +
engineOb.engineVersion());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```



```
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeDBEngineVersions](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Beschreiben der Optionen für DB-Instances

Das folgende Codebeispiel zeigt, wie Optionen für Aurora-DB-Instances beschrieben werden.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .engine("aurora-mysql")
            .defaultOnly(true)
            .maxRecords(20)
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engineOb : engines) {
            System.out.println("The name of the DB parameter group family for
the database engine is "
                + engineOb.dbParameterGroupFamily());
            System.out.println("The name of the database engine " +
engineOb.engine());
            System.out.println("The version number of the database engine " +
engineOb.engineVersion());
        }
    }
}
```


```
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}
```

- Weitere API-Informationen finden Sie unter [DescribeOrderableDBInstanceOptions](#) in der APIAWS SDK for Java 2.x -Referenz für .

Beschreiben von Parametern von einer DB-Cluster-Parametergruppe aus

Das folgende Codebeispiel zeigt, wie Parameter aus einer Aurora-DB-Cluster-Parametergruppe beschrieben werden.

SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void describeDbClusterParameters(RdsClient rdsClient, String  
dbClusterGroupName, int flag) {  
    try {  
        DescribeDbClusterParametersRequest dbParameterGroupsRequest;  
        if (flag == 0) {  
            dbParameterGroupsRequest =  
DescribeDbClusterParametersRequest.builder()  
                .dbClusterParameterGroupName(dbClusterGroupName)  
                .build();  
        } else {  
            dbParameterGroupsRequest =  
DescribeDbClusterParametersRequest.builder()  
                .dbClusterParameterGroupName(dbClusterGroupName)  
                .source("user")  
                .build();  
        }  
    }  
}
```

```

DescribeDbClusterParametersResponse response = rdsClient
    .describeDBClusterParameters(dbParameterGroupsRequest);
List<Parameter> dbParameters = response.parameters();
String paraName;
for (Parameter para : dbParameters) {
    // Only print out information about either auto_increment_offset or
    // auto_increment_increment.
    paraName = para.parameterName();
    if ((paraName.compareTo("auto_increment_offset") == 0)
        || (paraName.compareTo("auto_increment_increment ") == 0)) {
        System.out.println("*** The parameter name is " + paraName);
        System.out.println("*** The parameter value is " +
para.parameterValue());
        System.out.println("*** The parameter data type is " +
para.dataType());
        System.out.println("*** The parameter description is " +
para.description());
        System.out.println("*** The parameter allowed values is " +
para.allowedValues());
    }
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

```

- Weitere API-Informationen finden Sie unter [DescribeDBClusterParameters](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Aktualisieren von Parametern in einer DB-Cluster-Parametergruppe

Das folgende Codebeispiel zeigt, wie Parameter in einer Aurora-DB-Cluster-Parametergruppe aktualisiert werden.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
    try {
        DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .maxRecords(20)
            .build();

        List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
            .dbClusterParameterGroups();
        for (DBClusterParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbClusterParameterGroupName());
            System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [ModifyDBClusterParameterGroup](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Szenarien

### Erste Schritte mit DB-Clustern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine benutzerdefinierte Aurora-DB-Cluster-Parametergruppe 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 Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
 * services-use-secrets_RS.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets available engine families for Amazon Aurora MySQL-Compatible Edition
 * by calling the DescribeDbEngineVersions(Engine='aurora-mysql') method.
 * 2. Selects an engine family and creates a custom DB cluster parameter group
```

```

* by invoking the describeDBClusterParameters method.
* 3. Gets the parameter groups by invoking the describeDBClusterParameterGroups
* method.
* 4. Gets parameters in the group by invoking the describeDBClusterParameters
* method.
* 5. Modifies the auto_increment_offset parameter by invoking the
* modifyDbClusterParameterGroupRequest method.
* 6. Gets and displays the updated parameters.
* 7. Gets a list of allowed engine versions by invoking the
* describeDbEngineVersions method.
* 8. Creates an Aurora DB cluster database cluster that contains a MySQL
* database.
* 9. Waits for DB instance to be ready.
* 10. Gets a list of instance classes available for the selected engine.
* 11. Creates a database instance in the cluster.
* 12. Waits for DB instance to be ready.
* 13. Creates a snapshot.
* 14. Waits for DB snapshot to be ready.
* 15. Deletes the DB cluster.
* 16. Deletes the DB cluster group.
*/
public class AuroraScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
            "Usage:\n" +
            "    <dbClusterGroupName> <dbParameterGroupFamily>
<dbInstanceClusterIdentifier> <dbInstanceIdentifier> <dbName>
<dbSnapshotIdentifier><secretName>"
            +
            "Where:\n" +
            "    dbClusterGroupName - The name of the DB cluster parameter
group. \n" +
            "    dbParameterGroupFamily - The DB cluster parameter group family
name (for example, aurora-mysql5.7). \n"
            +
            "    dbInstanceClusterIdentifier - The instance cluster identifier
value.\n" +
            "    dbInstanceIdentifier - The database instance identifier.\n" +
            "    dbName - The database name.\n" +
            "    dbSnapshotIdentifier - The snapshot identifier.\n" +

```

```
        "    secretName - The name of the AWS Secrets Manager secret that
contains the database credentials\\\"\\n";
    ;

    if (args.length != 7) {
        System.out.println(usage);
        System.exit(1);
    }

    String dbClusterGroupName = args[0];
    String dbParameterGroupFamily = args[1];
    String dbInstanceClusterIdentifier = args[2];
    String dbInstanceIdentifier = args[3];
    String dbName = args[4];
    String dbSnapshotIdentifier = args[5];
    String secretName = args[6];

    // Retrieve the database credentials using AWS Secrets Manager.
    Gson gson = new Gson();
    User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
    String username = user.getUsername();
    String userPassword = user.getPassword();

    Region region = Region.US_WEST_2;
    RdsClient rdsClient = RdsClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon Aurora example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Return a list of the available DB engines");
    describeDBEngines(rdsClient);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Create a custom parameter group");
    createDBClusterParameterGroup(rdsClient, dbClusterGroupName,
dbParameterGroupFamily);
    System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("3. Get the parameter group");
describeDbClusterParameterGroups(rdsClient, dbClusterGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Get the parameters in the group");
describeDbClusterParameters(rdsClient, dbClusterGroupName, 0);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Modify the auto_increment_offset parameter");
modifyDBClusterParas(rdsClient, dbClusterGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Display the updated parameter value");
describeDbClusterParameters(rdsClient, dbClusterGroupName, -1);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Get a list of allowed engine versions");
getAllowedEngines(rdsClient, dbParameterGroupFamily);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Create an Aurora DB cluster database");
String arnClusterVal = createDBCluster(rdsClient, dbClusterGroupName,
dbName, dbInstanceClusterIdentifier,
    username, userPassword);
System.out.println("The ARN of the cluster is " + arnClusterVal);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Wait for DB instance to be ready");
waitForInstanceReady(rdsClient, dbInstanceClusterIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get a list of instance classes available for the
selected engine");
String instanceClass = getListInstanceClasses(rdsClient);
System.out.println(DASHES);
```



```
System.out.println(DASHES);
System.out.println("11. Create a database instance in the cluster.");
String clusterDBARN = createDBInstanceCluster(rdsClient,
dbInstanceIdentifier, dbInstanceClusterIdentifier,
instanceClass);
System.out.println("The ARN of the database is " + clusterDBARN);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Wait for DB instance to be ready");
waitDBInstanceReady(rdsClient, dbInstanceIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Create a snapshot");
createDBClusterSnapshot(rdsClient, dbInstanceClusterIdentifier,
dbSnapshotIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Wait for DB snapshot to be ready");
waitForSnapshotReady(rdsClient, dbSnapshotIdentifier,
dbInstanceClusterIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Delete the DB instance");
deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("15. Delete the DB cluster");
deleteCluster(rdsClient, dbInstanceClusterIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("16. Delete the DB cluster group");
deleteDBClusterGroup(rdsClient, dbClusterGroupName, clusterDBARN);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The Scenario has successfully completed.");
System.out.println(DASHES);
rdsClient.close();
```

```
}

private static SecretsManagerClient getSecretClient() {
    Region region = Region.US_WEST_2;
    return SecretsManagerClient.builder()
        .region(region)

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

private static String getSecretValues(String secretName) {
    SecretsManagerClient secretClient = getSecretClient();
    GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
        .secretId(secretName)
        .build();

    GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
    return valueResponse.secretString();
}

public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
    throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            didFind = false;
            int index = 1;
            for (DBInstance instance : instanceList) {
                instanceARN = instance.dbInstanceArn();
                if (instanceARN.compareTo(clusterDBARN) == 0) {
                    System.out.println(clusterDBARN + " still exists");
                    didFind = true;
                }
            }
        }
    }
}
```

```
        if ((index == listSize) && (!didFind)) {
            // Went through the entire list and did not find the
database ARN.
            isDataDel = true;
        }
        Thread.sleep(sleepTime * 1000);
        index++;
    }
}

DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
    .builder()
    .dbClusterParameterGroupName(dbClusterGroupName)
    .build();

rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
System.out.println(dbClusterGroupName + " was deleted.");

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
    try {
        DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .skipFinalSnapshot(true)
            .build();

        rdsClient.deleteDBCluster(deleteDbClusterRequest);
        System.out.println(dbInstanceClusterIdentifier + " was deleted!");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

```
public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .deleteAutomatedBackups(true)
            .skipFinalSnapshot(true)
            .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
String dbInstanceClusterIdentifier) {
    try {
        boolean snapshotReady = false;
        String snapshotReadyStr;
        System.out.println("Waiting for the snapshot to become available.");

        DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
            .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .build();

        while (!snapshotReady) {
            DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
            List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
            for (DBClusterSnapshot snapshot : snapshotList) {
                snapshotReadyStr = snapshot.status();
                if (snapshotReadyStr.contains("available")) {
                    snapshotReady = true;
                }
            }
        }
    }
}
```

```
        } else {
            System.out.println(".");
            Thread.sleep(sleepTime * 5000);
        }
    }
}

System.out.println("The Snapshot is available!");

} catch (RdsException | InterruptedException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
    String dbSnapshotIdentifier) {
    try {
        CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
        System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void waitDBInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");
    try {
        DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
```

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

    String endpoint = "";
    while (!instanceReady) {
        DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
        List<DBInstance> instanceList = response.dbInstances();
        for (DBInstance instance : instanceList) {
            instanceReadyStr = instance.dbInstanceStatus();
            if (instanceReadyStr.contains("available")) {
                endpoint = instance.endpoint().address();
                instanceReady = true;
            } else {
                System.out.print(".");
                Thread.sleep(sleepTime * 1000);
            }
        }
    }
    System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

    } catch (RdsException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static String createDBInstanceCluster(RdsClient rdsClient,
    String dbInstanceIdentifier,
    String dbInstanceClusterIdentifier,
    String instanceClass) {
    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .engine("aurora-mysql")
            .dbInstanceClass(instanceClass)
            .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
```

```
        System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
        return response.dbInstance().dbInstanceArn();

    } catch (RdsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static String getListInstanceClasses(RdsClient rdsClient) {
    try {
        DescribeOrderableDbInstanceOptionsRequest optionsRequest =
DescribeOrderableDbInstanceOptionsRequest
            .builder()
            .engine("aurora-mysql")
            .maxRecords(20)
            .build();

        DescribeOrderableDbInstanceOptionsResponse response = rdsClient
            .describeOrderableDBInstanceOptions(optionsRequest);
        List<OrderableDBInstanceOption> instanceOptions =
response.orderableDBInstanceOptions();
        String instanceClass = "";
        for (OrderableDBInstanceOption instanceOption : instanceOptions) {
            instanceClass = instanceOption.dbInstanceClass();
            System.out.println("The instance class is " +
instanceOption.dbInstanceClass());
            System.out.println("The engine version is " +
instanceOption.engineVersion());
        }
        return instanceClass;

    } catch (RdsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
```

```
boolean instanceReady = false;
String instanceReadyStr;
System.out.println("Waiting for instance to become available.");
try {
    DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
        .dbClusterIdentifier(dbClusterIdentifier)
        .build();

    while (!instanceReady) {
        DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
        List<DBCluster> clusterList = response.dbClusters();
        for (DBCluster cluster : clusterList) {
            instanceReadyStr = cluster.status();
            if (instanceReadyStr.contains("available")) {
                instanceReady = true;
            } else {
                System.out.print(".");
                Thread.sleep(sleepTime * 1000);
            }
        }
    }
    System.out.println("Database cluster is available!");

} catch (RdsException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

}

public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
    String dbClusterIdentifier, String userName, String password) {
    try {
        CreateDbClusterRequest clusterRequest = CreateDbClusterRequest.builder()
            .databaseName(dbName)
            .dbClusterIdentifier(dbClusterIdentifier)
            .dbClusterParameterGroupName(dbParameterGroupFamily)
            .engine("aurora-mysql")
            .masterUsername(userName)
            .masterUserPassword(password)
            .build();
```



```
        CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
        return response.dbCluster().dbClusterArn();

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
    try {
        DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .engine("aurora-mysql")
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
        List<DBEngineVersion> dbEngines = response.dbEngineVersions();
        for (DBEngineVersion dbEngine : dbEngines) {
            System.out.println("The engine version is " +
dbEngine.engineVersion());
            System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Modify the auto_increment_offset parameter.
public static void modifyDBClusterParas(RdsClient rdsClient, String
dClusterGroupName) {
    try {
        Parameter parameter1 = Parameter.builder()
            .parameterName("auto_increment_offset")
            .applyMethod("immediate")
```

```
        .parameterValue("5")
        .build();

    List<Parameter> paraList = new ArrayList<>();
    paraList.add(parameter1);
    ModifyDbClusterParameterGroupRequest groupRequest =
ModifyDbClusterParameterGroupRequest.builder()
        .dbClusterParameterGroupName(dClusterGroupName)
        .parameters(paraList)
        .build();

    ModifyDbClusterParameterGroupResponse response =
rdsClient.modifyDBClusterParameterGroup(groupRequest);
    System.out.println(
        "The parameter group " + response.dbClusterParameterGroupName()
+ " was successfully modified");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDbClusterParameters(RdsClient rdsClient, String
dbClusterGroupName, int flag) {
    try {
        DescribeDbClusterParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .build();
        } else {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .source("user")
                .build();
        }

        DescribeDbClusterParametersResponse response = rdsClient
            .describeDBClusterParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
```

```
        for (Parameter para : dbParameters) {
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            paraName = para.parameterName();
            if ((paraName.compareTo("auto_increment_offset") == 0)
                || (paraName.compareTo("auto_increment_increment ") == 0)) {
                System.out.println("*** The parameter name is " + paraName);
                System.out.println("*** The parameter value is " +
para.parameterValue());
                System.out.println("*** The parameter data type is " +
para.dataType());
                System.out.println("*** The parameter description is " +
para.description());
                System.out.println("*** The parameter allowed values is " +
para.allowedValues());
            }
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
    try {
        DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .maxRecords(20)
            .build();

        List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
            .dbClusterParameterGroups();
        for (DBClusterParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbClusterParameterGroupName());
            System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
        }

    } catch (RdsException e) {
```

```
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
        String dbParameterGroupFamily) {
    try {
        CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .description("Created by using the AWS SDK for Java")
            .build();

        CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .engine("aurora-mysql")
            .defaultOnly(true)
            .maxRecords(20)
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engineOb : engines) {
            System.out.println("The name of the DB parameter group family for
the database engine is ")

```

```
        + engine0b.dbParameterGroupFamily());
        System.out.println("The name of the database engine " +
engine0b.engine());
        System.out.println("The version number of the database engine " +
engine0b.engineVersion());
    }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [CreateDBCluster](#)
  - [CreateDBClusterParameterGroup](#)
  - [CreateDBClusterSnapshot](#)
  - [CreateDBInstance](#)
  - [DeleteDBCluster](#)
  - [DeleteDBClusterParameterGroup](#)
  - [DeleteDBInstance](#)
  - [DescribeDBClusterParameterGroups](#)
  - [DescribeDBClusterParameters](#)
  - [DescribeDBClusterSnapshots](#)
  - [DescribeDBClusters](#)
  - [DescribeDBEngineVersions](#)
  - [DescribeDBInstances](#)
  - [DescribeOrderableDBInstanceOptions](#)
  - [ModifyDBClusterParameterGroup](#)

## Auto Scaling-Beispiele mit SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Auto Scaling verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

## Erste Schritte

### Hallo Auto Scaling

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Auto Scaling.

#### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAutoScalingGroups {
    public static void main(String[] args) throws InterruptedException {
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    describeGroups(autoScalingClient);
}

public static void describeGroups(AutoScalingClient autoScalingClient) {
    DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups();
    List<AutoScalingGroup> groups = response.autoScalingGroups();
    groups.forEach(group -> {
        System.out.println("Group Name: " + group.autoScalingGroupName());
        System.out.println("Group ARN: " + group.autoScalingGroupARN());
    });
}
}
```

- Weitere API-Informationen finden Sie unter [DescribeAutoScalingGroups](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Themen

- [Aktionen](#)
- [Szenarien](#)

## Aktionen

### Erstellen einer Gruppe

Das folgende Codebeispiel zeigt, wie Sie eine Auto Scaling-Gruppe erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
    software.amazon.awssdk.services.autoscaling.model.CreateAutoScalingGroupRequest;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
    software.amazon.awssdk.services.autoscaling.model.LaunchTemplateSpecification;
import software.amazon.awssdk.services.autoscaling.waiters.AutoScalingWaiter;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <groupName> <launchTemplateName> <serviceLinkedRoleARN>
<vpcZoneId>

            Where:
                groupName - The name of the Auto Scaling group.
                launchTemplateName - The name of the launch template.\s
                vpcZoneId - A subnet Id for a virtual private cloud (VPC) where
instances in the Auto Scaling group can be created.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        String launchTemplateName = args[1];
        String vpcZoneId = args[2];
```



```
AutoScalingClient autoScalingClient = AutoScalingClient.builder()
    .region(Region.US_EAST_1)
    .build();

createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
vpcZoneId);
autoScalingClient.close();
}

public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
    String groupName,
    String launchTemplateName,
    String vpcZoneId) {

    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .availabilityZones("us-east-1a")
            .launchTemplate(templateSpecification)
            .maxSize(1)
            .minSize(1)
            .vpcZoneIdentifier(vpcZoneId)
            .build();

        autoScalingClient.createAutoScalingGroup(request);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
            .waitUntilGroupExists(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Auto Scaling Group created");

    } catch (AutoScalingException e) {
```

```

        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Weitere API-Informationen finden Sie unter [CreateAutoScalingGroup](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Löschen einer Gruppe

Das folgende Codebeispiel zeigt, wie Sie eine Auto Scaling-Gruppe löschen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
software.amazon.awssdk.services.autoscaling.model.DeleteAutoScalingGroupRequest;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = ""

        Usage:

```

```
        <groupName>

        Where:
            groupName - The name of the Auto Scaling group.
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String groupName = args[0];
    AutoScalingClient autoScalingClient = AutoScalingClient.builder()
        .region(Region.US_EAST_1)
        .build();

    deleteAutoScalingGroup(autoScalingClient, groupName);
    autoScalingClient.close();
}

public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .forceDelete(true)
            .build();

        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println("You successfully deleted " + groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [DeleteAutoScalingGroup](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Deaktivieren der Erfassung von Metriken für eine Gruppe

Das folgende Codebeispiel zeigt, wie Sie die Erfassung von CloudWatch Metriken für eine Auto Scaling-Gruppe deaktivieren.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .build();

        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
        System.out.println("The disable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DisableMetricsCollection](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Aktivieren der Erfassung von Metriken für eine Gruppe

Das folgende Codebeispiel zeigt, wie Sie die Erfassung von CloudWatch Metriken für eine Auto Scaling-Gruppe aktivieren.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .granularity("1Minute")
            .build();

        autoScalingClient.enableMetricsCollection(collectionRequest);
        System.out.println("The enable metrics collection operation was
successful");
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [EnableMetricsCollection](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Abrufen von Informationen zu Gruppen

Das folgende Codebeispiel zeigt, wie Sie Informationen zu Auto Scaling-Gruppen abrufen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import software.amazon.awssdk.services.autoscaling.model.Instance;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAutoScalingInstances {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <groupName>

                Where:
                groupName - The name of the Auto Scaling group.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String groupName = args[0];
AutoScalingClient autoScalingClient = AutoScalingClient.builder()
    .region(Region.US_EAST_1)
    .build();

String instanceId = getAutoScaling(autoScalingClient, groupName);
System.out.println(instanceId);
autoScalingClient.close();
}

public static String getAutoScaling(AutoScalingClient autoScalingClient, String
groupName) {
    try {
        String instanceId = "";
        DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        DescribeAutoScalingGroupsResponse response = autoScalingClient
            .describeAutoScalingGroups(scalingGroupsRequest);
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        for (AutoScalingGroup group : groups) {
            System.out.println("The group name is " +
group.autoScalingGroupName());
            System.out.println("The group ARN is " +
group.autoScalingGroupARN());

            List<Instance> instances = group.instances();
            for (Instance instance : instances) {
                instanceId = instance.instanceId();
            }
        }
        return instanceId;
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Weitere API-Informationen finden Sie unter [DescribeAutoScalingGroups](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Abrufen von Informationen zu Instances

Das folgende Codebeispiel zeigt, wie Sie Informationen zu Auto Scaling-Instances abrufen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
    try {
        DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest
= DescribeAutoScalingInstancesRequest
        .builder()
        .instanceIds(id)
        .build();

        DescribeAutoScalingInstancesResponse response = autoScalingClient
        .describeAutoScalingInstances(describeAutoScalingInstancesRequest);
        List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
        for (AutoScalingInstanceDetails instance : instances) {
            System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```



- Weitere API-Informationen finden Sie unter [DescribeAutoScalingInstances](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Informationen zu Skalierungsaktivitäten abrufen

Das folgende Codebeispiel zeigt, wie Sie Informationen zu Auto Scaling-Aktivitäten abrufen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
            .autoScalingGroupName(groupName)
            .maxRecords(10)
            .build();

        DescribeScalingActivitiesResponse response = autoScalingClient
            .describeScalingActivities(scalingActivitiesRequest);
        List<Activity> activities = response.activities();
        for (Activity activity : activities) {
            System.out.println("The activity Id is " + activity.activityId());
            System.out.println("The activity details are " +
activity.details());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeScalingActivities](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Festlegen der gewünschten Kapazität einer Gruppe

Das folgende Codebeispiel zeigt, wie Sie die gewünschte Kapazität einer Auto Scaling-Gruppe festlegen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
        .autoScalingGroupName(groupName)
        .desiredCapacity(2)
        .build();

        autoScalingClient.setDesiredCapacity(capacityRequest);
        System.out.println("You have set the DesiredCapacity to 2");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [SetDesiredCapacity](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Beenden einer Instance in einer Gruppe

Das folgende Codebeispiel zeigt, wie Sie eine Instance in einer Auto Scaling-Gruppe beenden.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
    try {
        TerminateInstanceInAutoScalingGroupRequest request =
        TerminateInstanceInAutoScalingGroupRequest.builder()
            .instanceId(instanceId)
            .shouldDecrementDesiredCapacity(false)
            .build();

        autoScalingClient.terminateInstanceInAutoScalingGroup(request);
        System.out.println("You have terminated instance " + instanceId);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [TerminateInstanceInAutoScalingGroup](#) in der APIAWS SDK for Java 2.x -Referenz zu .

## Aktualisieren einer Gruppe

Das folgende Codebeispiel zeigt, wie Sie die Konfiguration für eine Auto Scaling-Gruppe aktualisieren.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName,
    String launchTemplateName) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
            .maxSize(3)
            .autoScalingGroupName(groupName)
            .launchTemplate(templateSpecification)
            .build();

        autoScalingClient.updateAutoScalingGroup(groupRequest);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
            .waitUntilGroupInService(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("You successfully updated the auto scaling group " +
groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- Weitere API-Informationen finden Sie unter [UpdateAutoScalingGroup](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Szenarien

### Erstellen und Verwalten eines ausfallsicheren Services

Das folgende Codebeispiel zeigt, wie Sie einen Webservice mit Load Balancing erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Antwort des Webserver auf Anfragen und Zustandsprüfungen, indem Sie die AWS Systems Manager Parameter aktualisieren.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
public class Main {
```

```
    public static final String fileName = "C:\\AWS\\resworkflow\\
\\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\\
\\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\\
\\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\\
\\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException, InterruptedException
    {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and Manage
a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
    }
}
```

```
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
            This concludes the demo of how to build and manage a resilient
service.

            To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
            that were created for this demo.
            """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                Okay, we'll leave the resources intact.
                Don't forget to delete them when you're done with them or you
might incur unexpected charges.
            """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }
```

```
// Deletes the AWS resources used in this example.
private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
    throws IOException, InterruptedException {
    loadBalancer.deleteLoadBalancer(lbName);
    System.out.println("*** Wait 30 secs for resource to be deleted");
    TimeUnit.SECONDS.sleep(30);
    loadBalancer.deleteTargetGroup(targetGroupName);
    autoScaler.deleteAutoScalingGroup(autoScalingGroupName);
    autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
    autoScaler.deleteTemplate(templateName);
    database.deleteTable(tableName);
}

private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
    Scanner in = new Scanner(System.in);
    System.out.println(
        """
            For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources
            to set up a load-balanced web service endpoint and explore
some ways to make it resilient
            against various kinds of failures.

            Some of the resources create by this demo are:
            \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
            \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.
            \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
            \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
        """);

    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Creating and populating a DynamoDB table named " +
tableName);
```



```
Database database = new Database();
database.createTable(tableName, fileName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("""
    Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
    This script starts a Python web server defined in the `server.py`
script. The web server
    listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
    For demo purposes, this server is run as the root user. In
production, the best practice is to
    run a web server, such as Apache, with least-privileged credentials.

    The template also defines an IAM policy that each instance uses to
assume a role that grants
    permissions to access the DynamoDB recommendation table and Systems
Manager parameters
    that control the flow of the demo.
    """);

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(
    "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
System.out.println("*** Wait 30 secs for the VPC to be created");
TimeUnit.SECONDS.sleep(30);
AutoScaler autoScaler = new AutoScaler();
String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

System.out.println("""
    At this point, you have EC2 instances created. Once each instance
starts, it listens for
    HTTP requests. You can see these instances in the console or
continue with the demo.
    Press Enter when you're ready to continue.
```

```
        """);

    in.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Creating variables that control the flow of the demo.");
    ParameterHelper paramHelper = new ParameterHelper();
    paramHelper.reset();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("""
        Creating an Elastic Load Balancing target group and load balancer.
The target group
        defines how the load balancer connects to instances. The load
balancer provides a
        single endpoint where clients connect and dispatches requests to
instances in the group.
        """);

    String vpcId = autoScaler.getDefaultVPC();
    List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
    System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
    String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
    String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
    autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
    System.out.println("Verifying access to the load balancer endpoint...");
    boolean wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
    if (!wasSuccessul) {
        System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to "http://checkip.amazonaws.com"
        HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
        try {
            // Execute the request and get the response
            HttpResponse response = httpClient.execute(httpGet);
```

```
        // Read the response content.
        String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

        // Print the public IP address.
        System.out.println("Public IP Address: " + ipAddress);
        GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
        if (!groupInfo.isPortOpen()) {
            System.out.println("""
                For this example to work, the default security group for
your default VPC must
                allow access from this computer. You can either add it
automatically from this
                example or add it yourself using the AWS Management
Console.
                """);

            System.out.println(
                "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
            System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
            String ans = in.nextLine();
            if ("y".equalsIgnoreCase(ans)) {
                autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                System.out.println("Security group rule added.");
            } else {
                System.out.println("No security group rule added.");
            }
        }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
} else if (wasSuccessful) {
    System.out.println("Your load balancer is ready. You can access it by
browsing to:");
    System.out.println("\t http://" + elbDnsName);
} else {
    System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
```

```
        System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
        System.out.println("you can successfully make a GET request to the load
balancer.");
    }

    System.out.println("Press Enter when you're ready to continue with the
demo.");
    in.nextLine();
}

// A method that controls the demo part of the Java program.
public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    ParameterHelper paramHelper = new ParameterHelper();
    System.out.println("Read the ssm_only_policy.json file");
    String ssmOnlyPolicy = readFileAsString(ssmJSON);

    System.out.println("Resetting parameters to starting values for demo.");
    paramHelper.reset();

    System.out.println(
        """"
                This part of the demonstration shows how to toggle
different parts of the system
                to create situations where the web service fails, and shows
how using a resilient
                architecture can keep the web service running in spite of
these failures.

                At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
        """);
    demoChoices(loadBalancer);

    System.out.println(
        """"
                The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                The table name is contained in a Systems Manager parameter
named self.param_helper.table.
                To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
        """);
}
```

```
paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

System.out.println(
    ""
    \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
    healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
    "");
demoChoices(loadBalancer);

System.out.println(
    ""
    Instead of failing when the recommendation service fails,
the web service can return a static response.
    While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
    "");
paramHelper.put(paramHelper.failureResponse, "static");

System.out.println("""
    Now, sending a GET request to the load balancer endpoint returns a
static response.
    The service still reports as healthy because health checks are still
shallow.
    """);
demoChoices(loadBalancer);

System.out.println("Let's reinstate the recommendation service.");
paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

System.out.println("""
    Let's also substitute bad credentials for one of the instances in
the target group so that it can't
    access the DynamoDB recommendation table. We will get an instance id
value.
    """);

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
AutoScaler autoScaler = new AutoScaler();

// Create a new instance profile based on badCredsProfileName.
templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
```

```
String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
System.out.println("The association Id value is " + profileAssociationId);
System.out.println("Replacing the profile for instance " + badInstanceId
+ " with a profile that contains bad credentials");
autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

System.out.println(
    """"
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
        depending on which instance is selected by the load
balancer.
    """);

demoChoices(loadBalancer);

System.out.println("""
    Let's implement a deep health check. For this demo, a deep health
check tests whether
    the web service can access the DynamoDB table that it depends on for
recommendations. Note that
    the deep health check is only for ELB routing and not for Auto
Scaling instance health.
    This kind of deep health check is not recommended for Auto Scaling
instance health, because it
    risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
    """);

System.out.println("""
    By implementing deep health checks, the load balancer can detect
when one of the instances is failing
    and take that instance out of rotation.
    """);

paramHelper.put(paramHelper.healthCheck, "deep");

System.out.println("""
```

```
        Now, checking target health indicates that the instance with bad
credentials
        is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
        instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
        the load balancer takes unhealthy instances out of its rotation.
        """);

        demoChoices(loadBalancer);

        System.out.println(
            ""
            Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
            instance is to terminate it and let the auto scaler start a
new instance to replace it.
            """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
            Even while the instance is terminating and the new instance is
starting, sending a GET
            request to the web service continues to get a successful
recommendation response because
            the load balancer routes requests to the healthy instances. After
the replacement instance
            starts and reports as healthy, it is included in the load balancing
rotation.

            Note that terminating and replacing an instance typically takes
several minutes, during which time you
            can see the changing health check status until the new instance is
running and healthy.
            """);

        demoChoices(loadBalancer);
        System.out.println(
            "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }
}
```

```
public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    String[] actions = {
        "Send a GET request to the load balancer endpoint.",
        "Check the health of load balancer targets.",
        "Go to the next part of the demo."
    };
    Scanner scanner = new Scanner(System.in);

    while (true) {
        System.out.println("-".repeat(88));
        System.out.println("See the current state of the service by selecting
one of the following choices:");
        for (int i = 0; i < actions.length; i++) {
            System.out.println(i + ": " + actions[i]);
        }

        try {
            System.out.print("\nWhich action would you like to take? ");
            int choice = scanner.nextInt();
            System.out.println("-".repeat(88));

            switch (choice) {
                case 0 -> {
                    System.out.println("Request:\n");
                    System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                    CloseableHttpClient httpClient =
HttpClients.createDefault();

                    // Create an HTTP GET request to the ELB.
                    HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                    // Execute the request and get the response.
                    HttpResponse response = httpClient.execute(httpGet);
                    int statusCode = response.getStatusLine().getStatusCode();
                    System.out.println("HTTP Status Code: " + statusCode);

                    // Display the JSON response
                    BufferedReader reader = new BufferedReader(
                        new
InputStreamReader(response.getEntity().getContent()));
```



```

        StringBuilder jsonResponse = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            jsonResponse.append(line);
        }
        reader.close();

        // Print the formatted JSON response.
        System.out.println("Full Response:\n");
        System.out.println(jsonResponse.toString());

        // Close the HTTP client.
        httpClient.close();
    }
    case 1 -> {
        System.out.println("\nChecking the health of load balancer
targets:\n");
        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                                target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println("""
Note that it can take a minute or two for the health
check to update
                                after changes are made.
                                """);
    }
    case 2 -> {
        System.out.println("\nOkay, let's move on.");
        System.out.println("-".repeat(88));
        return; // Exit the method when choice is 2
    }
    default -> System.out.println("You must choose a value between
0-2. Please select again.");
}

} catch (java.util.InputMismatchException e) {
    System.out.println("Invalid input. Please select again.");
    scanner.nextLine(); // Clear the input buffer.
}

```

```
    }  
  }  
}  
  
public static String readFileAsString(String filePath) throws IOException {  
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));  
    return new String(bytes);  
}  
}
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
public class AutoScaler {  
  
    private static Ec2Client ec2Client;  
    private static AutoScalingClient autoScalingClient;  
    private static IamClient iamClient;  
  
    private static SsmClient ssmClient;  
  
    private IamClient getIAMClient() {  
        if (iamClient == null) {  
            iamClient = IamClient.builder()  
                .region(Region.US_EAST_1)  
                .build();  
        }  
        return iamClient;  
    }  
  
    private SsmClient getSSMClient() {  
        if (ssmClient == null) {  
            ssmClient = SsmClient.builder()  
                .region(Region.US_EAST_1)  
                .build();  
        }  
        return ssmClient;  
    }  
  
    private Ec2Client getEc2Client() {  
        if (ec2Client == null) {  
            ec2Client = Ec2Client.builder()  
                .region(Region.US_EAST_1)
```

```
        .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance is
 * terminated, it can no longer be accessed.
 */
public void terminateInstance(String instanceId) {
    TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
    TerminateInstanceInAutoScalingGroupRequest
        .builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

    getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
    System.out.format("Terminated instance %s.", instanceId);
}

/**
 * Replaces the profile associated with a running instance. After the profile is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
 * When
 * the instance is ready, Systems Manager is used to restart the Python web
 * server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
    throws InterruptedException {
    // Create an IAM instance profile specification.
```

```
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
    .builder()
    .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
                                // name.
    .build();

// Replace the IAM instance profile association for the EC2 instance.
ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
    .builder()
    .iamInstanceProfile(iamInstanceProfile)
    .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
    .build();

try {
    getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
    // Handle the response as needed.
} catch (Ec2Exception e) {
    // Handle exceptions, log, or report the error.
    System.err.println("Error: " + e.getMessage());
}

System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
    newInstanceProfileName);
TimeUnit.SECONDS.sleep(15);
boolean instReady = false;
int tries = 0;

// Reboot after 60 seconds
while (!instReady) {
    if (tries % 6 == 0) {
        getEc2Client().rebootInstances(RebootInstancesRequest.builder()
            .instanceIds(instanceId)
            .build());
        System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
    }
    tries++;
    try {
        TimeUnit.SECONDS.sleep(10);
    }
}
```

```

    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
    List<InstanceInformation> instanceInformationList =
informationResponse.getInstanceInformationList();
    for (InstanceInformation info : instanceInformationList) {
        if (info.getInstanceId().equals(instanceId)) {
            instReady = true;
            break;
        }
    }
}

SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
    .instanceIds(instanceId)
    .documentName("AWS-RunShellScript")
    .parameters(Collections.singletonMap("commands",
80")))
        Collections.singletonList("cd / && sudo python3 server.py

    .build();

getSSMClient().sendCommand(sendCommandRequest);
System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
}

public void openInboundPort(String secGroupId, String port, String ipAddress) {
    AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
        .groupName(secGroupId)
        .cidrIp(ipAddress)
        .fromPort(Integer.parseInt(port))
        .build();

    getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
    System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
}

/**
 * Detaches a role from an instance profile, detaches policies from the role,

```

```
    * and deletes all the resources.
    */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
            getInstanceProfileRequest =
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                .builder()
                .instanceProfileName(profileName)
                .build();

            GetInstanceProfileResponse response =
            getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
            RemoveRoleFromInstanceProfileRequest.builder()
                .instanceProfileName(profileName)
                .roleName(roleName)
                .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
            DeleteInstanceProfileRequest.builder()
                .instanceProfileName(profileName)
                .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                .roleName(roleName)
                .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
            rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(attachedPolicy.policyArn())
```

```
        .build();

        getIAMClient().detachRolePolicy(request);
        System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
    }

    getIAMClient().deleteRole(deleteRoleRequest);
    System.out.println("Instance profile and role deleted.");

} catch (IamException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
    System.out.format(templateName + " was deleted.");
}

public void deleteAutoScaleGroup(String groupName) {
    DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
        .autoScalingGroupName(groupName)
        .forceDelete(true)
        .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println(groupName + " was deleted.");
}

/*
 * Verify the default security group of the specified VPC allows ingress from
 * this
 * computer. This can be done by allowing ingress from this computer's IP
 * address. In some situations, such as connecting from a corporate network, you
 * must instead specify a prefix list ID. You can also temporarily open the port
 * to
 * any IP address while running this example. If you do, be sure to remove
 * public
 * access when you're done.
 */
```

```
*
*/
public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
    boolean portIsOpen = false;
    GroupInfo groupInfo = new GroupInfo();
    try {
        Filter filter = Filter.builder()
            .name("group-name")
            .values("default")
            .build();

        Filter filter1 = Filter.builder()
            .name("vpc-id")
            .values(VPC)
            .build();

        DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
            .filters(filter, filter1)
            .build();

        DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
            .describeSecurityGroups(securityGroupsRequest);
        String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
        groupInfo.setGroupName(securityGroup);

        for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
            System.out.println("Found security group: " + secGroup.groupId());

            for (IpPermission ipPermission : secGroup.ipPermissions()) {
                if (ipPermission.fromPort() == port) {
                    System.out.println("Found inbound rule: " + ipPermission);
                    for (IpRange ipRange : ipPermission.ipRanges()) {
                        String cidrIp = ipRange.cidrIp();
                        if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                            System.out.println(cidrIp + " is applicable");
                            portIsOpen = true;
                        }
                    }
                }

                if (!ipPermission.prefixListIds().isEmpty()) {
                    System.out.println("Prefix lList is applicable");
                }
            }
        }
    }
}
```



```
        portIsOpen = true;
    }

    if (!portIsOpen) {
        System.out
            .println("The inbound rule does not appear to be
open to either this computer's IP,"
                    + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
    } else {
        break;
    }
}
}

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}

groupInfo.setPortOpen(portIsOpen);
return groupInfo;
}

/**
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {
        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
            .autoScalingGroupName(asGroupName)
            .targetGroupARNs(targetGroupARN)
            .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
        System.out.println("Attached load balancer to " + asGroupName);
    }
}
```

```
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

    // Get availability zones.
    software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
    .builder()
    .build();

    DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
    List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
    .collect(Collectors.toList());

    String availabilityZones = String.join(",", availabilityZoneNames);
    LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
        .launchTemplateName(templateName)
        .version("$Default")
        .build();

    String[] zones = availabilityZones.split(",");
    CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
        .launchTemplate(specification)
        .availabilityZones(zones)
        .maxSize(groupSize)
        .minSize(groupSize)
        .autoScalingGroupName(autoScalingGroupName)
        .build();

    try {
        getAutoScalingClient().createAutoScalingGroup(groupRequest);
    }
```

```
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
    return zones;
}

public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();

    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
        .builder()
        .filters(defaultFilter)
        .build();

    DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
    return response.vpcs().get(0).vpcId();
}

// Gets the default subnets in a VPC for a specified list of Availability Zones.
public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
    List<Subnet> subnets = null;
    Filter vpcFilter = Filter.builder()
        .name("vpc-id")
        .values(vpcId)
        .build();

    Filter azFilter = Filter.builder()
        .name("availability-zone")
        .values(availabilityZones)
        .build();

    Filter defaultForAZ = Filter.builder()
        .name("default-for-az")
        .values("true")
        .build();
```

```
DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
    .filters(vpcFilter, azFilter, defaultForAZ)
    .build();

DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
subnets = response.subnets();
return subnets;
}

// Gets data about the instances in the EC2 Auto Scaling group.
public String getBadInstance(String groupName) {
    DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
    .autoScalingGroupNames(groupName)
    .build();

    DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
    AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
    List<String> instanceIds = autoScalingGroup.instances().stream()
        .map(instance -> instance.instanceId())
        .collect(Collectors.toList());

    String[] instanceIdArray = instanceIds.toArray(new String[0]);
    for (String instanceId : instanceIdArray) {
        System.out.println("Instance ID: " + instanceId);
        return instanceId;
    }
    return "";
}

// Gets data about the profile associated with an instance.
public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();
```

```

        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
            .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                .builder()
                .policyArn(policy.arn())
                .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                    .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                    || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .roleName(roleName) // Specify the name of the IAM role
                        .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
                    .policyArn(policy.arn())
                    .build();

```

```
        getIAMClient().deletePolicy(deletePolicyRequest);
        System.out.println("Policy deleted successfully.");
        break;
    }
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
    RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(InstanceProfile)
        .roleName(roleName) // Remove the extra dot here
        .build();

    getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
    System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
}

// Delete the instance profile after removing all roles
DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
    .instanceProfileName(InstanceProfile)
    .build();

getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
System.out.println(InstanceProfile + " Deleted");
System.out.println("All roles and policies are deleted.");
}
}
```

Erstellen Sie eine Klasse, die Elastic-Load-Balancing-Aktionen beinhaltet.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
        DescribeTargetGroupsRequest.builder()
            .names(targetGroupName)
            .build();

        DescribeTargetGroupsResponse tgResponse =
        getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
        DescribeTargetHealthRequest.builder()
            .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
            .build();

        DescribeTargetHealthResponse healthResponse =
        getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
```

```

    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {

```



```
boolean success = false;
int retries = 3;
CloseableHttpClient httpClient = HttpClients.createDefault();

// Create an HTTP GET request to the ELB.
HttpGet httpGet = new HttpGet("http://" + elbDnsName);
try {
    while ((!success) && (retries > 0)) {
        // Execute the request and get the response.
        HttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        System.out.println("HTTP Status Code: " + statusCode);
        if (statusCode == 200) {
            success = true;
        } else {
            retries--;
            System.out.println("Got connection error from load balancer
endpoint, retrying...");
            TimeUnit.SECONDS.sleep(15);
        }
    }

} catch (org.apache.http.conn.HttpHostConnectException e) {
    System.out.println(e.getMessage());
}

System.out.println("Status.." + success);
return success;
}

/*
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
```

```
        .name(targetGroupName)
        .protocol(protocol)
        .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /**
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
        String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                .map(Subnet::subnetId)
                .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                .subnets(subnetIdStrings)
                .name(lbName)
                .scheme("internet-facing")
                .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
```

```
        .loadBalancerArns(lbARN)
        .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " + e.getMessage());
        }
        return false;
    }

    /**
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended, such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
```

```
* MediaType,
* forms a unique identifier for the recommended item.
*/
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("ItemId")
                    .attributeType(ScalarAttributeType.N)
                    .build())
            .keySchema(
                KeySchemaElement.builder()
                    .attributeName("MediaType")
                    .keyType(KeyType.HASH)
                    .build(),
                KeySchemaElement.builder()
                    .attributeName("ItemId")
                    .keyType(KeyType.RANGE)
                    .build())
            .provisionedThroughput(
                ProvisionedThroughput.builder()
                    .readCapacityUnits(5L)
                    .writeCapacityUnits(5L)
                    .build())
            .build();

        getDynamoDbClient().createTable(createTableRequest);
        System.out.println("Creating table " + tableName + "...");

        // Wait until the Amazon DynamoDB table is created.
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();
    }
}
```

```
        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Table " + tableName + " created.");

        // Add records to the table.
        populateTable(fileName, tableName);
    }
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.
public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
}
```

```
        System.out.println("Added all records to the " + tableName);
    }
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
            .overwrite(true)
            .type("String")
            .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [AttachLoadBalancerTargetGroups](#)
  - [CreateAutoScalingGroup](#)
  - [CreateInstanceProfile](#)

- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacesIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

## Verwalten von Gruppen und Instances

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Amazon EC2 Auto Scaling-Gruppe mit einer Startvorlage und Availability Zones und erhalten Sie Informationen über laufende Instances.
- Aktivieren Sie die Erfassung von Amazon- CloudWatch Metriken.
- ~~Aktualisieren Sie die gewünschte Kapazität der Gruppe und warten Sie, bis eine Instance gestartet wird.~~



- Beenden Sie eine Instance in der Gruppe.
- Auflisten von Skalierungsaktivitäten, die als Reaktion auf Benutzeranfragen und Kapazitätsänderungen auftreten.
- Rufen Sie Statistiken für CloudWatch Metriken ab und bereinigen Sie dann Ressourcen.

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, create a launch template. For more information, see the
 * following topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-
 * templates.html#create-launch-template
 *
 * This code example performs the following operations:
 * 1. Creates an Auto Scaling group using an AutoScalingWaiter.
 * 2. Gets a specific Auto Scaling group and returns an instance Id value.
 * 3. Describes Auto Scaling with the Id value.
 * 4. Enables metrics collection.
 * 5. Update an Auto Scaling group.
 * 6. Describes Account details.
 * 7. Describe account details"
 * 8. Updates an Auto Scaling group to use an additional instance.
 * 9. Gets the specific Auto Scaling group and gets the number of instances.
 * 10. List the scaling activities that have occurred for the group.
 * 11. Terminates an instance in the Auto Scaling group.
 * 12. Stops the metrics collection.
 * 13. Deletes the Auto Scaling group.
```

```
*/

public class AutoScalingScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

            Usage:
                <groupName> <launchTemplateName> <vpcZoneId>

            Where:
                groupName - The name of the Auto Scaling group.
                launchTemplateName - The name of the launch template.\s
                vpcZoneId - A subnet Id for a virtual private cloud (VPC) where
instances in the Auto Scaling group can be created.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        String launchTemplateName = args[1];
        String vpcZoneId = args[2];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EC2 Auto Scaling example
scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an Auto Scaling group named " + groupName);
        createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
vpcZoneId);
        System.out.println(
            "Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);
    }
}
```

```
System.out.println(DASHES);
System.out.println("2. Get Auto Scale group Id value");
String instanceId = getSpecificAutoScalingGroups(autoScalingClient,
groupNames);
if (instanceId.compareTo("") == 0) {
    System.out.println("Error - no instance Id value");
    System.exit(1);
} else {
    System.out.println("The instance Id value is " + instanceId);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Describe Auto Scaling with the Id value " +
instanceId);
describeAutoScalingInstance(autoScalingClient, instanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Enable metrics collection " + instanceId);
enableMetricsCollection(autoScalingClient, groupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Update an Auto Scaling group to update max size to
3");
updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Describe Auto Scaling groups");
describeAutoScalingGroups(autoScalingClient, groupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Describe account details");
describeAccountLimits(autoScalingClient);
System.out.println(
    "Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned");
Thread.sleep(60000);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("8. Set desired capacity to 2");
setDesiredCapacity(autoScalingClient, groupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Get the two instance Id values and state");
getSpecificAutoScalingGroups(autoScalingClient, groupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. List the scaling activities that have occurred for
the group");
describeScalingActivities(autoScalingClient, groupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Terminate an instance in the Auto Scaling group");
terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Stop the metrics collection");
disableMetricsCollection(autoScalingClient, groupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Delete the Auto Scaling group");
deleteAutoScalingGroup(autoScalingClient, groupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The Scenario has successfully completed.");
System.out.println(DASHES);

autoScalingClient.close();
}

public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
            .autoScalingGroupName(groupName)
```

```
        .maxRecords(10)
        .build();

    DescribeScalingActivitiesResponse response = autoScalingClient
        .describeScalingActivities(scalingActivitiesRequest);
    List<Activity> activities = response.activities();
    for (Activity activity : activities) {
        System.out.println("The activity Id is " + activity.activityId());
        System.out.println("The activity details are " +
activity.details());
    }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

    public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
            .autoScalingGroupName(groupName)
            .desiredCapacity(2)
            .build();

        autoScalingClient.setDesiredCapacity(capacityRequest);
        System.out.println("You have set the DesiredCapacity to 2");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

    public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName,
String launchTemplateName,
String vpcZoneId) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
```

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

    CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
        .autoScalingGroupName(groupName)
        .availabilityZones("us-east-1a")
        .launchTemplate(templateSpecification)
        .maxSize(1)
        .minSize(1)
        .vpcZoneIdentifier(vpcZoneId)
        .build();

    autoScalingClient.createAutoScalingGroup(request);
    DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

    WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
        .waitUntilGroupExists(groupsRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Auto Scaling Group created");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
    try {
        DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest
= DescribeAutoScalingInstancesRequest
            .builder()
            .instanceIds(id)
            .build();

        DescribeAutoScalingInstancesResponse response = autoScalingClient
            .describeAutoScalingInstances(describeAutoScalingInstancesRequest);
```

```
        List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
        for (AutoScalingInstanceDetails instance : instances) {
            System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .maxRecords(10)
            .build();

        DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        for (AutoScalingGroup group : groups) {
            System.out.println("*** The service to use for the health checks: "
+ group.healthCheckType());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String getSpecificAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        String instanceId = "";
        DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();
```

```
DescribeAutoScalingGroupsResponse response = autoScalingClient
    .describeAutoScalingGroups(ScalingGroupsRequest);
List<AutoScalingGroup> groups = response.autoScalingGroups();
for (AutoScalingGroup group : groups) {
    System.out.println("The group name is " +
group.autoScalingGroupName());
    System.out.println("The group ARN is " +
group.autoScalingGroupARN());
    List<Instance> instances = group.instances();

    for (Instance instance : instances) {
        instanceId = instance.instanceId();
        System.out.println("The instance id is " + instanceId);
        System.out.println("The lifecycle state is " +
instance.lifecycleState());
    }
}

return instanceId;
} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .granularity("1Minute")
            .build();

        autoScalingClient.enableMetricsCollection(collectionRequest);
        System.out.println("The enable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```



```
    }  
  }  
  
  public static void disableMetricsCollection(AutoScalingClient autoScalingClient,  
String groupName) {  
    try {  
      DisableMetricsCollectionRequest disableMetricsCollectionRequest =  
DisableMetricsCollectionRequest.builder()  
        .autoScalingGroupName(groupName)  
        .metrics("GroupMaxSize")  
        .build();  
  
autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);  
      System.out.println("The disable metrics collection operation was  
successful");  
  
    } catch (AutoScalingException e) {  
      System.err.println(e.awsErrorDetails().errorMessage());  
      System.exit(1);  
    }  
  }  
  
  public static void describeAccountLimits(AutoScalingClient autoScalingClient) {  
    try {  
      DescribeAccountLimitsResponse response =  
autoScalingClient.describeAccountLimits();  
      System.out.println("The max number of auto scaling groups is " +  
response.maxNumberOfAutoScalingGroups());  
      System.out.println("The current number of auto scaling groups is " +  
response.numberOfWorkAutoScalingGroups());  
  
    } catch (AutoScalingException e) {  
      System.err.println(e.awsErrorDetails().errorMessage());  
      System.exit(1);  
    }  
  }  
  
  public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,  
String groupName,  
String launchTemplateName) {  
    try {  
      AutoScalingWaiter waiter = autoScalingClient.waiter();
```

```
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
        .launchTemplateName(launchTemplateName)
        .build();

        UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
        .maxSize(3)
        .autoScalingGroupName(groupName)
        .launchTemplate(templateSpecification)
        .build();

        autoScalingClient.updateAutoScalingGroup(groupRequest);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
        .waitUntilGroupInService(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("You successfully updated the auto scaling group " +
groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
    try {
        TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

        autoScalingClient.terminateInstanceInAutoScalingGroup(request);
        System.out.println("You have terminated instance " + instanceId);

    } catch (AutoScalingException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .forceDelete(true)
            .build();

        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println("You successfully deleted " + groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [CreateAutoScalingGroup](#)
  - [DeleteAutoScalingGroup](#)
  - [DescribeAutoScalingGroups](#)
  - [DescribeAutoScalingInstances](#)
  - [DescribeScalingActivities](#)
  - [DisableMetricsCollection](#)
  - [EnableMetricsCollection](#)
  - [SetDesiredCapacity](#)
  - [TerminateInstanceInAutoScalingGroup](#)
  - [UpdateAutoScalingGroup](#)

## Amazon-Bedrock-Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon Bedrock verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)

### Aktionen

Abrufen von Details zu einem Amazon-Bedrock-Grundlagenmodell

Das folgende Codebeispiel zeigt, wie Sie Details zu einem Amazon-Bedrock-Grundlagenmodell abrufen.

SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Abrufen von Details zu einem Grundlagenmodell mithilfe des synchronen Amazon-Bedrock-Clients.

```
/**
 * Get details about an Amazon Bedrock foundation model.
 *
 * @param bedrockClient The service client for accessing Amazon Bedrock.
```

```
    * @param modelIdentifier The model identifier.
    * @return An object containing the foundation model's details.
    */
    public static FoundationModelDetails getFoundationModel(BedrockClient
bedrockClient, String modelIdentifier) {
        try {
            GetFoundationModelResponse response = bedrockClient.getFoundationModel(
                r -> r.modelIdentifier(modelIdentifier)
            );

            FoundationModelDetails model = response.modelDetails();

            System.out.println(" Model ID:                " + model.modelId());
            System.out.println(" Model ARN:                " +
model.modelArn());
            System.out.println(" Model Name:                " +
model.modelName());
            System.out.println(" Provider Name:            " +
model.providerName());
            System.out.println(" Lifecycle status:        " +
model.modelLifecycle().statusAsString());
            System.out.println(" Input modalities:        " +
model.inputModalities());
            System.out.println(" Output modalities:       " +
model.outputModalities());
            System.out.println(" Supported customizations: " +
model.customizationsSupported());
            System.out.println(" Supported inference types: " +
model.inferenceTypesSupported());
            System.out.println(" Response streaming supported: " +
model.responseStreamingSupported());

            return model;

        } catch (ValidationException e) {
            throw new IllegalArgumentException(e.getMessage());
        } catch (SdkException e) {
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    }
}
```

Abrufen von Details zu einem Grundlagenmodell mithilfe des asynchronen Amazon-Bedrock-Clients.

```
/**
 * Get details about an Amazon Bedrock foundation model.
 *
 * @param bedrockClient The async service client for accessing Amazon Bedrock.
 * @param modelIdentifier The model identifier.
 * @return An object containing the foundation model's details.
 */
public static FoundationModelDetails getFoundationModel(BedrockAsyncClient
bedrockClient, String modelIdentifier) {
    try {
        CompletableFuture<GetFoundationModelResponse> future =
bedrockClient.getFoundationModel(
            r -> r.modelIdentifier(modelIdentifier)
        );

        FoundationModelDetails model = future.get().modelDetails();

        System.out.println(" Model ID: " + model.modelId());
        System.out.println(" Model ARN: " +
model.modelArn());
        System.out.println(" Model Name: " +
model.modelName());
        System.out.println(" Provider Name: " +
model.providerName());
        System.out.println(" Lifecycle status: " +
model.modelLifecycle().statusAsString());
        System.out.println(" Input modalities: " +
model.inputModalities());
        System.out.println(" Output modalities: " +
model.outputModalities());
        System.out.println(" Supported customizations: " +
model.customizationsSupported());
        System.out.println(" Supported inference types: " +
model.inferenceTypesSupported());
        System.out.println(" Response streaming supported: " +
model.responseStreamingSupported());

        return model;
    } catch (ExecutionException e) {
```

```

        if (e.getMessage().contains("ValidationException")) {
            throw new IllegalArgumentException(e.getMessage());
        } else {
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    }
}

```

- Weitere API-Informationen finden Sie unter [GetFoundationModel](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Auflisten verfügbarer Amazon-Bedrock-Grundlagenmodelle

Das folgende Codebeispiel zeigt, wie Sie verfügbare Amazon-Bedrock-Grundlagenmodelle auflisten.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Listen Sie die verfügbaren Amazon-Bedrock-Grundlagenmodelle mit dem synchronen Amazon-Bedrock-Client auf.

```

/**
 * Lists Amazon Bedrock foundation models that you can use.
 * You can filter the results with the request parameters.
 *
 * @param bedrockClient The service client for accessing Amazon Bedrock.
 * @return A list of objects containing the foundation models' details
 */
public static List<FoundationModelSummary> listFoundationModels(BedrockClient
bedrockClient) {

```

```

    try {
        ListFoundationModelsResponse response =
bedrockClient.listFoundationModels(r -> {});

        List<FoundationModelSummary> models = response.modelSummaries();

        if (models.isEmpty()) {
            System.out.println("No available foundation models in " +
region.toString());
        } else {
            for (FoundationModelSummary model : models) {
                System.out.println("Model ID: " + model.modelId());
                System.out.println("Provider: " + model.providerName());
                System.out.println("Name:      " + model.modelName());
                System.out.println();
            }
        }

        return models;

    } catch (SdkClientException e) {
        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    }
}

```

Listen Sie die verfügbaren Amazon-Bedrock-Grundlagenmodelle mit dem asynchronen Amazon-Bedrock-Client auf.

```

/**
 * Lists Amazon Bedrock foundation models that you can use.
 * You can filter the results with the request parameters.
 *
 * @param bedrockClient The async service client for accessing Amazon Bedrock.
 * @return A list of objects containing the foundation models' details
 */
public static List<FoundationModelSummary>
listFoundationModels(BedrockAsyncClient bedrockClient) {
    try {
        CompletableFuture<ListFoundationModelsResponse> future =
bedrockClient.listFoundationModels(r -> {});

```



```
        List<FoundationModelSummary> models = future.get().modelSummaries();

        if (models.isEmpty()) {
            System.out.println("No available foundation models in " +
region.toString());
        } else {
            for (FoundationModelSummary model : models) {
                System.out.println("Model ID: " + model.modelId());
                System.out.println("Provider: " + model.providerName());
                System.out.println("Name:      " + model.modelName());
                System.out.println();
            }
        }

        return models;

    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    } catch (ExecutionException e) {
        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    }
}
```

- Weitere API-Informationen finden Sie unter [ListFoundationModels](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Amazon-Bedrock-Laufzeitbeispiele mit SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon Bedrock Runtime verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

## Themen

- [Aktionen](#)
- [Szenarien](#)

## Aktionen

### Bildgenerierung mit Amazon Titan Image Generator G1

Das folgende Codebeispiel zeigt, wie das G1-Modell des Amazon Titan Image Generators auf Amazon Bedrock zur Bildgenerierung aufgerufen wird.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Rufen Sie asynchron das Modell Amazon Titan Image Generator G1 auf, um Bilder zu generieren.

```
/**
 * Invokes the Amazon Titan image generation model to create an image using the
 * input
 * provided in the request body.
 *
 * @param prompt The prompt that you want Amazon Titan to use for image
 * generation.
 * @param seed The random noise seed for image generation (Range: 0 to
 * 2147483647).
 * @return A Base64-encoded string representing the generated image.
 */
public static String invokeTitanImage(String prompt, long seed) {
    /**
     * The different model providers have individual request and response
     formats.
     * For the format, ranges, and default values for Titan Image models refer
     to:
```

```
titan-
    * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-titan-
    \* image.html
    */
String titanImageModelId = "amazon.titan-image-generator-v1";

BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

var textToImageParams = new JSONObject().put("text", prompt);

var imageGenerationConfig = new JSONObject()
    .put("numberOfImages", 1)
    .put("quality", "standard")
    .put("cfgScale", 8.0)
    .put("height", 512)
    .put("width", 512)
    .put("seed", seed);

JSONObject payload = new JSONObject()
    .put("taskType", "TEXT_IMAGE")
    .put("textToImageParams", textToImageParams)
    .put("imageGenerationConfig", imageGenerationConfig);

InvokeModelRequest request = InvokeModelRequest.builder()
    .body(SdkBytes.fromUtf8String(payload.toString()))
    .modelId(titanImageModelId)
    .contentType("application/json")
    .accept("application/json")
    .build();

CompletableFuture<InvokeModelResponse> completableFuture =
client.invokeModel(request)
    .whenComplete((response, exception) -> {
        if (exception != null) {
            System.out.println("Model invocation failed: " + exception);
        }
    });

String base64ImageData = "";
try {
    InvokeModelResponse response = completableFuture.get();
```

```
        JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
        base64ImageData = responseBody
            .getJSONArray("images")
            .getString(0);

    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        System.err.println(e.getMessage());
    } catch (ExecutionException e) {
        System.err.println(e.getMessage());
    }

    return base64ImageData;
}
```

Rufen Sie das G1-Modell des Amazon Titan Image Generators auf, um Bilder zu generieren.

```
/**
 * Invokes the Amazon Titan image generation model to create an image using
the
 * input
 * provided in the request body.
 *
 * @param prompt The prompt that you want Amazon Titan to use for image
 * generation.
 * @param seed The random noise seed for image generation (Range: 0 to
 * 2147483647).
 * @return A Base64-encoded string representing the generated image.
 */
public static String invokeTitanImage(String prompt, long seed) {
    /**
 * The different model providers have individual request and
response formats.
 * For the format, ranges, and default values for Titan Image models
refer to:
 * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-titan-
 \* image.html
 */
    String titanImageModelId = "amazon.titan-image-generator-v1";
```

```
        BedrockRuntimeClient client = BedrockRuntimeClient.builder()
            .region(Region.US_EAST_1)

.credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        var textToImageParams = new JSONObject().put("text", prompt);

        var imageGenerationConfig = new JSONObject()
            .put("numberOfImages", 1)
            .put("quality", "standard")
            .put("cfgScale", 8.0)
            .put("height", 512)
            .put("width", 512)
            .put("seed", seed);

        JSONObject payload = new JSONObject()
            .put("taskType", "TEXT_IMAGE")
            .put("textToImageParams", textToImageParams)
            .put("imageGenerationConfig",
imageGenerationConfig);

        InvokeModelRequest request = InvokeModelRequest.builder()
            .body(SdkBytes.fromUtf8String(payload.toString()))
            .modelId(titanImageModelId)
            .contentType("application/json")
            .accept("application/json")
            .build();

        InvokeModelResponse response = client.invokeModel(request);

        JSONObject responseBody = new
JSONObject(response.body().asUtf8String());

        String base64ImageData = responseBody
            .getJSONArray("images")
            .getString(0);

        return base64ImageData;
    }
}
```

- Weitere API-Informationen finden Sie unter [InvokeModel](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Bildgenerierung mit Stability.ai Stable XL

Das folgende Codebeispiel zeigt, wie das Modell Stability.ai Stable XL auf Amazon Bedrock zur Bildgenerierung aufgerufen wird.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Rufen Sie asynchron das Foundation-Modell Stability.ai Stable XL auf, um Bilder zu generieren.

```
/**
 * Asynchronously invokes the Stability.ai Stable Diffusion XL model to create
 * an image based on the provided input.
 *
 * @param prompt      The prompt that guides the Stable Diffusion model.
 * @param seed        The random noise seed for image generation (use 0 or omit
 *                    for a random seed).
 * @param stylePreset The style preset to guide the image model towards a
 *                    specific style.
 * @return A Base64-encoded string representing the generated image.
 */
public static String invokeStableDiffusion(String prompt, long seed, String
stylePreset) {
    /**
     * The different model providers have individual request and response
     formats.
     * For the format, ranges, and available style_presets of Stable Diffusion
     * models refer to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
     stability-diffusion.html
     */

    String stableDiffusionModelId = "stability.stable-diffusion-xl";
```

```
BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

JSONArray wrappedPrompt = new JSONArray().put(new JSONObject().put("text",
prompt));
JSONObject payload = new JSONObject()
    .put("text_prompts", wrappedPrompt)
    .put("seed", seed);

if (stylePreset != null && !stylePreset.isEmpty()) {
    payload.put("style_preset", stylePreset);
}

InvokeModelRequest request = InvokeModelRequest.builder()
    .body(SdkBytes.fromUtf8String(payload.toString()))
    .modelId(stableDiffusionModelId)
    .contentType("application/json")
    .accept("application/json")
    .build();

CompletableFuture<InvokeModelResponse> completableFuture =
client.invokeModel(request)
    .whenComplete((response, exception) -> {
        if (exception != null) {
            System.out.println("Model invocation failed: " + exception);
        }
    });

String base64ImageData = "";
try {
    InvokeModelResponse response = completableFuture.get();
    JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
    base64ImageData = responseBody
        .getJSONArray("artifacts")
        .getJSONObject(0)
        .getString("base64");
} catch (InterruptedException e) {
    Thread.currentThread().interrupt();
    System.err.println(e.getMessage());
} catch (ExecutionException e) {
```

```

        System.err.println(e.getMessage());
    }

    return base64ImageData;
}

```

Rufen Sie das Foundation-Modell Stability.ai Stable XL auf, um Bilder zu generieren.

```

/**
 * Invokes the Stability.ai Stable Diffusion XL model to create an image
based
 * on the provided input.
 *
 * @param prompt      The prompt that guides the Stable Diffusion model.
 * @param seed        The random noise seed for image generation (use 0 or
omit
 *                    for a random seed).
 * @param stylePreset The style preset to guide the image model towards a
 *                    specific style.
 * @return A Base64-encoded string representing the generated image.
 */
public static String invokeStableDiffusion(String prompt, long seed, String
stylePreset) {
    /**
     * The different model providers have individual request and
response formats.
     * For the format, ranges, and available style_presets of Stable
Diffusion
     * models refer to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-
parameters-stability-diffusion.html
     */

    String stableDiffusionModelId = "stability.stable-diffusion-xl";

    BedrockRuntimeClient client = BedrockRuntimeClient.builder()
        .region(Region.US_EAST_1)

        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

```



```
JSONArray wrappedPrompt = new JSONArray().put(new
JSONObject().put("text", prompt));

JSONObject payload = new JSONObject()
    .put("text_prompts", wrappedPrompt)
    .put("seed", seed);

if (!(stylePreset == null || stylePreset.isEmpty())) {
    payload.put("style_preset", stylePreset);
}

InvokeModelRequest request = InvokeModelRequest.builder()
    .body(SdkBytes.fromUtf8String(payload.toString()))
    .modelId(stableDiffusionModelId)
    .contentType("application/json")
    .accept("application/json")
    .build();

InvokeModelResponse response = client.invokeModel(request);

JSONObject responseBody = new
JSONObject(response.body().asUtf8String());

String base64ImageData = responseBody
    .getJSONArray("artifacts")
    .getJSONObject(0)
    .getString("base64");

return base64ImageData;
}
```

- Weitere API-Informationen finden Sie unter [InvokeModel](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Textgenerierung mit AI21 Labs Jurassic-2

Das folgende Codebeispiel zeigt, wie das AI21 Labs Jurassic-2-Modell auf Amazon Bedrock zur Textgenerierung aufgerufen wird.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Rufen Sie das AI21 Labs Jurassic-2-Grundlagenmodell asynchron auf, um Text zu generieren.

```
/**
 * Asynchronously invokes the AI21 Labs Jurassic-2 model to run an inference
 * based on the provided input.
 *
 * @param prompt The prompt that you want Jurassic to complete.
 * @return The inference response generated by the model.
 */
public static String invokeJurassic2(String prompt) {
    /**
     * The different model providers have individual request and response
    formats.
     * For the format, ranges, and default values for Anthropic Claude, refer
    to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
    claude.html
     */

    String jurassic2ModelId = "ai21.j2-mid-v1";

    BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
        .region(Region.US_EAST_1)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    String payload = new JSONObject()
        .put("prompt", prompt)
        .put("temperature", 0.5)
        .put("maxTokens", 200)
        .toString();

    InvokeModelRequest request = InvokeModelRequest.builder()
        .body(SdkBytes.fromUtf8String(payload))
        .modelId(jurassic2ModelId)
```

```

        .contentType("application/json")
        .accept("application/json")
        .build();

    CompletableFuture<InvokeModelResponse> completableFuture =
client.invokeModel(request)
        .whenComplete((response, exception) -> {
            if (exception != null) {
                System.out.println("Model invocation failed: " + exception);
            }
        });

    String generatedText = "";
    try {
        InvokeModelResponse response = completableFuture.get();
        JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
        generatedText = responseBody
            .getJSONArray("completions")
            .getJSONObject(0)
            .getJSONObject("data")
            .getString("text");

    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        System.err.println(e.getMessage());
    } catch (ExecutionException e) {
        System.err.println(e.getMessage());
    }

    return generatedText;
}

```

Rufen Sie das AI21 Labs Jurassic-2-Grundlagenmodell auf, um Text zu generieren.

```

/**
 * Invokes the AI21 Labs Jurassic-2 model to run an inference based on the
 * provided input.
 *
 * @param prompt The prompt for Jurassic to complete.
 * @return The generated response.
 */

```

```
public static String invokeJurassic2(String prompt) {
    /*
     * The different model providers have individual request and
response formats.
     * For the format, ranges, and default values for AI21 Labs
Jurassic-2, refer
     * to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-jurassic2.html
     */

    String jurassic2ModelId = "ai21.j2-mid-v1";

    BedrockRuntimeClient client = BedrockRuntimeClient.builder()
        .region(Region.US_EAST_1)

.credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    String payload = new JSONObject()
        .put("prompt", prompt)
        .put("temperature", 0.5)
        .put("maxTokens", 200)
        .toString();

    InvokeModelRequest request = InvokeModelRequest.builder()
        .body(SdkBytes.fromUtf8String(payload))
        .modelId(jurassic2ModelId)
        .contentType("application/json")
        .accept("application/json")
        .build();

    InvokeModelResponse response = client.invokeModel(request);

    JSONObject responseBody = new
JSONObject(response.body().asUtf8String());

    String generatedText = responseBody
        .getJSONArray("completions")
        .getJSONObject(0)
        .getJSONObject("data")
        .getString("text");

    return generatedText;
}
```

```
}

```

- Weitere API-Informationen finden Sie unter [InvokeModel](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Textgenerierung mit Anthropic Claude 2

Das folgende Codebeispiel zeigt, wie das Modell Anthropic Claude 2 in Amazon Bedrock zur Textgenerierung aufgerufen wird.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Rufen Sie das Grundlagenmodell Anthropic Claude 2 asynchron auf, um Text zu generieren.

```
/**
 * Asynchronously invokes the Anthropic Claude 2 model to run an inference based
 * on the provided input.
 *
 * @param prompt The prompt that you want Claude to complete.
 * @return The inference response from the model.
 */
public static String invokeClaude(String prompt) {
    /**
     * The different model providers have individual request and response
    formats.
     * For the format, ranges, and default values for Anthropic Claude, refer
    to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
    claude.html
     */

    String claudeModelId = "anthropic.claude-v2";

    // Claude requires you to enclose the prompt as follows:
    String enclosedPrompt = "Human: " + prompt + "\n\nAssistant:";

```

```
BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

String payload = new JSONObject()
    .put("prompt", enclosedPrompt)
    .put("max_tokens_to_sample", 200)
    .put("temperature", 0.5)
    .put("stop_sequences", List.of("\n\nHuman:"))
    .toString();

InvokeModelRequest request = InvokeModelRequest.builder()
    .body(SdkBytes.fromUtf8String(payload))
    .modelId(claudeModelId)
    .contentType("application/json")
    .accept("application/json")
    .build();

CompletableFuture<InvokeModelResponse> completableFuture =
client.invokeModel(request)
    .whenComplete((response, exception) -> {
        if (exception != null) {
            System.out.println("Model invocation failed: " + exception);
        }
    });

String generatedText = "";
try {
    InvokeModelResponse response = completableFuture.get();
    JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
    generatedText = responseBody.getString("completion");
} catch (InterruptedException e) {
    Thread.currentThread().interrupt();
    System.err.println(e.getMessage());
} catch (ExecutionException e) {
    System.err.println(e.getMessage());
}

return generatedText;
}
```

Rufen Sie das Grundlagenmodell Anthropic Claude 2 auf, um Text zu generieren.

```
/**
 * Invokes the Anthropic Claude 2 model to run an inference based on the
 * provided input.
 *
 * @param prompt The prompt for Claude to complete.
 * @return The generated response.
 */
public static String invokeClaude(String prompt) {
    /**
     * The different model providers have individual request and
response formats.
     * For the format, ranges, and default values for Anthropic Claude,
refer to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-claude.html
     */

    String claudeModelId = "anthropic.claude-v2";

    // Claude requires you to enclose the prompt as follows:
    String enclosedPrompt = "Human: " + prompt + "\n\nAssistant:";

    BedrockRuntimeClient client = BedrockRuntimeClient.builder()
        .region(Region.US_EAST_1)

        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    String payload = new JSONObject()
        .put("prompt", enclosedPrompt)
        .put("max_tokens_to_sample", 200)
        .put("temperature", 0.5)
        .put("stop_sequences", List.of("\n\nHuman:"))
        .toString();

    InvokeModelRequest request = InvokeModelRequest.builder()
        .body(SdkBytes.fromUtf8String(payload))
        .modelId(claudeModelId)
        .contentType("application/json")
}
```

```
        .accept("application/json")
        .build();

    InvokeModelResponse response = client.invokeModel(request);

    JSONObject responseBody = new
    JSONObject(response.body().asUtf8String());

    String generatedText = responseBody.getString("completion");

    return generatedText;
}
```

- Weitere API-Informationen finden Sie unter [InvokeModel](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Textgenerierung mit Anthropic Claude 2 mit einem Antwort-Stream

Das folgende Codebeispiel zeigt, wie das Modell Anthropic Claude 2 in Amazon Bedrock für die Textgenerierung mit einem Antwortstream aufgerufen wird.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Rufen Sie das Modell Anthropic Claude 2 auf und verarbeiten Sie den Antwort-Stream.

```
/**
 * Invokes the Anthropic Claude 2 model and processes the response stream.
 *
 * @param prompt The prompt for Claude to complete.
 * @param silent Suppress console output of the individual response stream
 *              chunks.
 * @return The generated response.
 */
public static String invokeClaude(String prompt, boolean silent) {
```



```
BedrockRuntimeAsyncClient client =
BedrockRuntimeAsyncClient.builder()
    .region(Region.US_EAST_1)

.credentialsProvider(ProfileCredentialsProvider.create())
    .build();

var finalCompletion = new AtomicReference<>("");

var payload = new JSONObject()
    .put("prompt", "Human: " + prompt + " Assistant:")
    .put("temperature", 0.8)
    .put("max_tokens_to_sample", 300)
    .toString();

var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(payload))
    .modelId("anthropic.claude-v2")
    .contentType("application/json")
    .accept("application/json")
    .build();

var visitor =
InvokeModelWithResponseStreamResponseHandler.Visitor.builder()
    .onChunk(chunk -> {
        var json = new
JSONObject(chunk.bytes().asUtf8String());
        var completion =
json.getString("completion");
        finalCompletion.set(finalCompletion.get() +
completion);

        if (!silent) {
            System.out.print(completion);
        }
    })
    .build();

var handler = InvokeModelWithResponseStreamResponseHandler.builder()
    .onEventStream(stream -> stream.subscribe(event ->
event.accept(visitor)))
    .onComplete(() -> {
    })
    .onError(e -> System.out.println("\n\nError: " +
e.getMessage()))
```

```

        .build();

        client.invokeModelWithResponseStream(request, handler).join();

        return finalCompletion.get();
    }

```

- Weitere API-Informationen finden Sie unter [InvokeModelWithResponseStream](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Textgenerierung mit Meta Llama 2 Chat

Das folgende Codebeispiel zeigt, wie das Meta-Llama-2-Chat-Modell auf Amazon Bedrock zur Textgenerierung aufgerufen wird.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Rufen Sie das Meta-Llama-2-Chat-Grundlagenmodell asynchron auf, um Text zu generieren.

```

/**
 * Asynchronously invokes the Meta Llama 2 Chat model to run an inference based
 * on the provided input.
 *
 * @param prompt The prompt that you want Llama 2 to complete.
 * @return The inference response generated by the model.
 */
public static String invokeLlama2(String prompt) {
    /**
     * The different model providers have individual request and response
    formats.
     * For the format, ranges, and default values for Meta Llama 2 Chat, refer
    to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
    meta.
     * html

```

```
    */

    String llama2ModelId = "meta.llama2-13b-chat-v1";

    BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
        .region(Region.US_EAST_1)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    String payload = new JSONObject()
        .put("prompt", prompt)
        .put("max_gen_len", 512)
        .put("temperature", 0.5)
        .put("top_p", 0.9)
        .toString();

    InvokeModelRequest request = InvokeModelRequest.builder()
        .body(SdkBytes.fromUtf8String(payload))
        .modelId( llama2ModelId)
        .contentType("application/json")
        .accept("application/json")
        .build();

    CompletableFuture<InvokeModelResponse> completableFuture =
client.invokeModel(request)
        .whenComplete((response, exception) -> {
            if (exception != null) {
                System.out.println("Model invocation failed: " + exception);
            }
        });

    String generatedText = "";
    try {
        InvokeModelResponse response = completableFuture.get();
        JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
        generatedText = responseBody.getString("generation");

    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        System.err.println(e.getMessage());
    } catch (ExecutionException e) {
        System.err.println(e.getMessage());
    }
}
```

```
        return generatedText;
    }
}
```

Rufen Sie das Meta-Llama-2-Chat-Grundlagenmodell auf, um Text zu generieren.

```
/**
 * Invokes the Meta Llama 2 Chat model to run an inference based on the
provided
 * input.
 *
 * @param prompt The prompt for Llama 2 to complete.
 * @return The generated response.
 */
public static String invokeLlama2(String prompt) {
    /**
     * The different model providers have individual request and
response formats.
     * For the format, ranges, and default values for Meta Llama 2 Chat,
refer to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-meta.html
     */

    String llama2ModelId = "meta.llama2-13b-chat-v1";

    BedrockRuntimeClient client = BedrockRuntimeClient.builder()
        .region(Region.US_EAST_1)

        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    String payload = new JSONObject()
        .put("prompt", prompt)
        .put("max_gen_len", 512)
        .put("temperature", 0.5)
        .put("top_p", 0.9)
        .toString();

    InvokeModelRequest request = InvokeModelRequest.builder()
        .body(SdkBytes.fromUtf8String(payload))
}
```

```

        .modelId(llama2ModelId)
        .contentType("application/json")
        .accept("application/json")
        .build();

    InvokeModelResponse response = client.invokeModel(request);

    JSONObject responseBody = new
    JSONObject(response.body().asUtf8String());

    String generatedText = responseBody.getString("generation");

    return generatedText;
}

```

- Weitere API-Informationen finden Sie unter [InvokeModel](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Textgenerierung mit Mistral 7B

Das folgende Codebeispiel zeigt, wie Sie das Modell Mistral 7B in Amazon Bedrock zur Textgenerierung aufrufen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Rufen Sie das Grundlagenmodell Mistral 7B asynchron auf, um Text zu generieren.

```

/**
 * Asynchronously invokes the Mistral 7B model to run an inference based on the
 * provided input.
 *
 * @param prompt The prompt for Mistral to complete.
 * @return The generated response.
 */
public static List<String> invokeMistral7B(String prompt) {

```

```
BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
    .region(Region.US_WEST_2)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

// Mistral instruct models provide optimal results when
// embedding the prompt into the following template:
String instruction = "<s>[INST] " + prompt + " [/INST]";

String modelId = "mistral.mistral-7b-instruct-v0:2";

String payload = new JSONObject()
    .put("prompt", instruction)
    .put("max_tokens", 200)
    .put("temperature", 0.5)
    .toString();

CompletableFuture<InvokeModelResponse> completableFuture =
client.invokeModel(request -> request
    .accept("application/json")
    .contentType("application/json")
    .body(SdkBytes.fromUtf8String(payload))
    .modelId(modelId))
    .whenComplete((response, exception) -> {
        if (exception != null) {
            System.out.println("Model invocation failed: " + exception);
        }
    });

try {
    InvokeModelResponse response = completableFuture.get();
    JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
    JSONArray outputs = responseBody.getJSONArray("outputs");

    return IntStream.range(0, outputs.length())
        .mapToObj(i -> outputs.getJSONObject(i).getString("text"))
        .toList();
} catch (InterruptedException e) {
    Thread.currentThread().interrupt();
    System.err.println(e.getMessage());
} catch (ExecutionException e) {
    System.err.println(e.getMessage());
}
```

```

    return List.of();
}

```

Rufen Sie das Grundlagenmodell Mistral 7B auf, um Text zu generieren.

```

/**
 * Invokes the Mistral 7B model to run an inference based on the provided
input.
 *
 * @param prompt The prompt for Mistral to complete.
 * @return The generated responses.
 */
public static List<String> invokeMistral7B(String prompt) {
    BedrockRuntimeClient client = BedrockRuntimeClient.builder()
        .region(Region.US_WEST_2)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    // Mistral instruct models provide optimal results when
    // embedding the prompt into the following template:
    String instruction = "<s>[INST] " + prompt + " [/INST]";

    String modelId = "mistral.mistral-7b-instruct-v0:2";

    String payload = new JSONObject()
        .put("prompt", instruction)
        .put("max_tokens", 200)
        .put("temperature", 0.5)
        .toString();

    InvokeModelResponse response = client.invokeModel(request -> request
        .accept("application/json")
        .contentType("application/json")
        .body(SdkBytes.fromUtf8String(payload))
        .modelId(modelId));

    JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
    JSONArray outputs = responseBody.getJSONArray("outputs");

    return IntStream.range(0, outputs.length())

```

```
        .mapToObj(i -> outputs.getJSONObject(i).getString("text"))
        .toList();
    }
}
```

- Weitere API-Informationen finden Sie unter [InvokeModel](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Textgenerierung mit Mixtral 8x7B

Das folgende Codebeispiel zeigt, wie das Modell Mixtral 8x7B in Amazon Bedrock zur Textgenerierung aufgerufen wird.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Rufen Sie asynchron das Grundlagenmodell Mistral 8x7B auf, um Text zu generieren.

```
/**
 * Asynchronously invokes the Mixtral 8x7B model to run an inference based on
 the provided input.
 *
 * @param prompt The prompt for Mixtral to complete.
 * @return The generated response.
 */
public static List<String> invokeMixtral8x7B(String prompt) {
    BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
        .region(Region.US_WEST_2)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    // Mistral instruct models provide optimal results when
    // embedding the prompt into the following template:
    String instruction = "<s>[INST] " + prompt + " [/INST]";
```



```

String modelId = "mistral.mixtral-8x7b-instruct-v0:1";

String payload = new JSONObject()
    .put("prompt", instruction)
    .put("max_tokens", 200)
    .put("temperature", 0.5)
    .toString();

CompletableFuture<InvokeModelResponse> completableFuture =
client.invokeModel(request -> request
    .accept("application/json")
    .contentType("application/json")
    .body(SdkBytes.fromUtf8String(payload))
    .modelId(modelId))
    .whenComplete((response, exception) -> {
        if (exception != null) {
            System.out.println("Model invocation failed: " + exception);
        }
    });

try {
    InvokeModelResponse response = completableFuture.get();
    JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
    JSONArray outputs = responseBody.getJSONArray("outputs");

    return IntStream.range(0, outputs.length())
        .mapToObj(i -> outputs.getJSONObject(i).getString("text"))
        .toList();
} catch (InterruptedException e) {
    Thread.currentThread().interrupt();
    System.err.println(e.getMessage());
} catch (ExecutionException e) {
    System.err.println(e.getMessage());
}

return List.of();
}

```

Rufen Sie das Grundlagenmodell Mixtral 8x7B auf, um Text zu generieren.

```
public static List<String> invokeMixtral8x7B(String prompt) {
```

```

BedrockRuntimeClient client = BedrockRuntimeClient.builder()
    .region(Region.US_WEST_2)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

// Mistral instruct models provide optimal results when
// embedding the prompt into the following template:
String instruction = "<s>[INST] " + prompt + " [/INST]";

String modelId = "mistral.mixtral-8x7b-instruct-v0:1";

String payload = new JSONObject()
    .put("prompt", instruction)
    .put("max_tokens", 200)
    .put("temperature", 0.5)
    .toString();

InvokeModelResponse response = client.invokeModel(request -> request
    .accept("application/json")
    .contentType("application/json")
    .body(SdkBytes.fromUtf8String(payload))
    .modelId(modelId));

JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
JSONArray outputs = responseBody.getJSONArray("outputs");

return IntStream.range(0, outputs.length())
    .mapToObj(i -> outputs.getJSONObject(i).getString("text"))
    .toList();
}

```

- Weitere API-Informationen finden Sie unter [InvokeModel](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Szenarien

Erstellen einer Playground-Anwendung für die Interaktion mit Amazon-Bedrock-Grundlagenmodellen

Das folgende Codebeispiel zeigt, wie Sie Playgrounds erstellen, um über verschiedene Modalitäten mit Amazon-Bedrock-Grundlagenmodellen zu interagieren.

## SDK für Java 2.x

Der Java Foundation Model (FM) Playground ist eine Spring Boot-Beispielanwendung, die zeigt, wie Amazon Bedrock mit Java verwendet wird. Dieses Beispiel zeigt, wie Java-Entwickler Amazon Bedrock verwenden können, um generative KI-fähige Anwendungen zu erstellen. Sie können die folgenden drei Playgrounds verwenden, um Amazon-Bedrock-Grundlagenmodelle zu testen und mit ihnen zu interagieren:

- Ein Text-Playground.
- Ein Chat-Playground.
- Ein Image-Playground.

Das Beispiel listet auch die Grundlagenmodelle auf, auf die Sie Zugriff haben, zusammen mit ihren Merkmalen. Quellcode und Bereitstellungsanweisungen finden Sie im Projekt unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- Amazon-Bedrock-Laufzeit

## CloudFront -Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit verwenden CloudFront.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)
- [Szenarien](#)

## Aktionen

### Eine Verteilung erstellen

Das folgende Codebeispiel zeigt, wie Sie eine CloudFront Verteilung erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Im folgenden Beispiel wird ein Amazon Simple Storage Service (Amazon S3)-Bucket als Inhaltsursprung verwendet.

Nach dem Erstellen der Verteilung erstellt der Code eine , [CloudFrontWaiter](#) um zu warten, bis die Verteilung bereitgestellt ist, bevor die Verteilung zurückgegeben wird.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreateDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.ItemSelection;
import software.amazon.awssdk.services.cloudfront.model.Method;
import software.amazon.awssdk.services.cloudfront.model.ViewerProtocolPolicy;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;
import software.amazon.awssdk.services.s3.S3Client;

import java.time.Instant;

public class CreateDistribution {

    private static final Logger logger =
        LoggerFactory.getLogger(CreateDistribution.class);

    public static Distribution createDistribution(CloudFrontClient
        cloudFrontClient, S3Client s3Client,
```

```

        final String bucketName, final String keyGroupId, final
String originAccessControlId) {

        final String region = s3Client.headBucket(b ->
b.bucket(bucketName)).sdkHttpResponse().headers()
                .get("x-amz-bucket-region").get(0);
        final String originDomain = bucketName + ".s3." + region +
".amazonaws.com";
        String originId = originDomain; // Use the originDomain value for
the originId.

        // The service API requires some deprecated methods, such as
// DefaultCacheBehavior.Builder#minTTL and #forwardedValue.
        CreateDistributionResponse createDistResponse =
cloudFrontClient.createDistribution(builder -> builder
                .distributionConfig(b1 -> b1
                        .origins(b2 -> b2
                                .quantity(1)
                                .items(b3 -> b3

                                .domainName(originDomain)

                                .id(originId)

                                .s3originConfig(builder4 -> builder4
                                        .originAccessIdentity(
                                                ""))

                                .originAccessControlId(
                                        originAccessControlId)))
                        .defaultCacheBehavior(b2 -> b2

                                .viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)

                                .targetOriginId(originId)

                                .minTTL(200L)
                                .forwardedValues(b5

                                -> b5

                                .cookies(cp -> cp

```

```

        .forward(ItemSelection.NONE))

    .queryString(true))
    -> b3

    .quantity(1)

    .items(keyGroupId)

    .enabled(true))
    > b4

    .quantity(2)

    .items(Method.HEAD, Method.GET)

    .cachedMethods(b5 -> b5

        .quantity(2)

        .items(Method.HEAD,

            Method.GET))))

    .cacheBehaviors(b -> b
        .quantity(1)
        .items(b2 -> b2

    .pathPattern("/index.html")

    .viewerProtocolPolicy(

        ViewerProtocolPolicy.ALLOW_ALL)

    .targetOriginId(originId)

    .trustedKeyGroups(b3 -> b3

        .quantity(1)

        .items(keyGroupId)

```

```

        .enabled(true))

    .minTTL(200L)

    .forwardedValues(b4 -> b4

        .cookies(cp -> cp

            .forward(ItemSelection.NONE))

        .queryString(true))

    .allowedMethods(b5 -> b5.quantity(2)

        .items(Method.HEAD,

            Method.GET)

        .cachedMethods(b6 -> b6

            .quantity(2)

            .items(Method.HEAD,

                Method.GET))))))
        .enabled(true)
        .comment("Distribution built with
java")

    .callerReference(Instant.now().toString()));

    final Distribution distribution = createDistResponse.distribution();
    logger.info("Distribution created. DomainName: [{}] Id: [{}]",
distribution.domainName(),
                distribution.id());
    logger.info("Waiting for distribution to be deployed ...");
    try (CloudFrontWaiter cfWaiter =
CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
        ResponseOrException<GetDistributionResponse>
responseOrException = cfWaiter
                .waitUntilDistributionDeployed(builder ->
builder.id(distribution.id()))
                .matched();

```

```

        responseOrException.response()
            .orElseThrow(() -> new
RuntimeException("Distribution not created"));
        logger.info("Distribution deployed. DomainName: [{}] Id:
[{}]", distribution.domainName(),
            distribution.id());
    }
    return distribution;
}
}

```

- Weitere API-Informationen finden Sie unter [CreateDistribution](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erstellen einer -Funktion

Das folgende Codebeispiel zeigt, wie Sie eine Amazon- CloudFront Funktion erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionRequest;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionResponse;
import software.amazon.awssdk.services.cloudfront.model.FunctionConfig;
import software.amazon.awssdk.services.cloudfront.model.FunctionRuntime;
import java.io.InputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:

```



```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateFunction {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <functionName> <filePath>

            Where:
                functionName - The name of the function to create.\s
                filePath - The path to a file that contains the application
logic for the function.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        String filePath = args[1];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
            .region(Region.AWS_GLOBAL)
            .build();

        String funArn = createNewFunction(cloudFrontClient, functionName, filePath);
        System.out.println("The function ARN is " + funArn);
        cloudFrontClient.close();
    }

    public static String createNewFunction(CloudFrontClient cloudFrontClient, String
functionName, String filePath) {
        try {
            InputStream fileIs =
CreateFunction.class.getClassLoader().getResourceAsStream(filePath);
            SdkBytes functionCode = SdkBytes.fromInputStream(fileIs);

            FunctionConfig config = FunctionConfig.builder()
                .comment("Created by using the CloudFront Java API")
                .runtime(FunctionRuntime.CLOUDFRONT_JS_1_0)
                .build();
```

```
        CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
            .name(functionName)
            .functionCode(functionCode)
            .functionConfig(config)
            .build();

        CreateFunctionResponse response =
cloudFrontClient.createFunction(functionRequest);
        return response.functionSummary().functionMetadata().functionARN();

    } catch (CloudFrontException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Weitere API-Informationen finden Sie unter [CreateFunction](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen einer Schlüsselgruppe

Das folgende Codebeispiel zeigt, wie Sie eine Schlüsselgruppe erstellen, die Sie mit signierten URLs und signierten Cookies verwenden können.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Eine Schlüsselgruppe benötigt mindestens einen öffentlichen Schlüssel, der zur Überprüfung signierter URLs oder Cookies verwendet wird.

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

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;

import java.util.UUID;

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

    public static String createKeyGroup(CloudFrontClient cloudFrontClient, String
publicKeyId) {
        String keyGroupId = cloudFrontClient.createKeyGroup(b -> b.keyGroupConfig(c
-> c
            .items(publicKeyId)
            .name("JavaKeyGroup" + UUID.randomUUID()))
            .keyGroup().id());
        logger.info("KeyGroup created with ID: [{}]", keyGroupId);
        return keyGroupId;
    }
}
```

- Weitere API-Informationen finden Sie unter [CreateKeyGroup](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen einer -Verteilung

Das folgende Codebeispiel zeigt, wie Sie eine CloudFront Verteilung löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Im folgenden Codebeispiel wird eine Verteilung auf deaktiviert aktualisiert, ein Waiter verwendet, der auf die Bereitstellung der Änderung wartet, und dann die Verteilung löscht.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
```

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;

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

    public static void deleteDistribution(final CloudFrontClient
cloudFrontClient, final String distributionId) {
        // First, disable the distribution by updating it.
        GetDistributionResponse response =
cloudFrontClient.getDistribution(b -> b
            .id(distributionId));
        String etag = response.eTag();
        DistributionConfig distConfig =
response.distribution().distributionConfig();

        cloudFrontClient.updateDistribution(builder -> builder
            .id(distributionId)
            .distributionConfig(builder1 -> builder1

.cacheBehaviors(distConfig.cacheBehaviors())

.defaultCacheBehavior(distConfig.defaultCacheBehavior())
                .enabled(false)
                .origins(distConfig.origins())
                .comment(distConfig.comment())

.callerReference(distConfig.callerReference())

.defaultCacheBehavior(distConfig.defaultCacheBehavior())
                .priceClass(distConfig.priceClass())
                .aliases(distConfig.aliases())
                .logging(distConfig.logging())

.defaultRootObject(distConfig.defaultRootObject())

.customErrorResponses(distConfig.customErrorResponses())

.httpVersion(distConfig.httpVersion())
```

```

.isIPV6Enabled(distConfig.isIPV6Enabled())

.restrictions(distConfig.restrictions())

.viewerCertificate(distConfig.viewerCertificate())
                    .webACLId(distConfig.webACLId())

.originGroups(distConfig.originGroups())
                .ifMatch(etag));

        logger.info("Distribution [{}] is DISABLED, waiting for deployment
before deleting ...",
                    distributionId);
        GetDistributionResponse distributionResponse;
        try (CloudFrontWaiter cfWaiter =
CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse>
responseOrException = cfWaiter
                        .waitUntilDistributionDeployed(builder ->
builder.id(distributionId)).matched();
            distributionResponse = responseOrException.response()
                .orElseThrow(() -> new
RuntimeException("Could not disable distribution"));
        }

        DeleteDistributionResponse deleteDistributionResponse =
cloudFrontClient
                .deleteDistribution(builder -> builder
                    .id(distributionId)

.isIfMatch(distributionResponse.eTag()));
        if (deleteDistributionResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Distribution [{}] DELETED", distributionId);
        }
    }
}

```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [DeleteDistribution](#)
  - [UpdateDistribution](#)

## Löschen von Signaturressourcen

Das folgende Codebeispiel zeigt, wie Sie Ressourcen löschen, die für den Zugriff auf eingeschränkte Inhalte in einem Amazon Simple Storage Service (Amazon S3)-Bucket verwendet werden.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteKeyGroupResponse;
import
    software.amazon.awssdk.services.cloudfront.model.DeleteOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.DeletePublicKeyResponse;
import software.amazon.awssdk.services.cloudfront.model.GetKeyGroupResponse;
import
    software.amazon.awssdk.services.cloudfront.model.GetOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.GetPublicKeyResponse;

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

    public static void deleteOriginAccessControl(final CloudFrontClient
        cloudFrontClient,
        final String originAccessControlId) {
        GetOriginAccessControlResponse getResponse = cloudFrontClient
            .getOriginAccessControl(b -> b.id(originAccessControlId));
        DeleteOriginAccessControlResponse deleteResponse =
            cloudFrontClient.deleteOriginAccessControl(builder -> builder
                .id(originAccessControlId)
                .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Origin Access Control [{}]",
                originAccessControlId);
        }
    }
}
```

```
    }

    public static void deleteKeyGroup(final CloudFrontClient cloudFrontClient, final
String keyGroupId) {

        GetKeyGroupResponse getResponse = cloudFrontClient.getKeyGroup(b ->
b.id(keyGroupId));
        DeleteKeyGroupResponse deleteResponse =
cloudFrontClient.deleteKeyGroup(builder -> builder
            .id(keyGroupId)
            .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Key Group [{}]", keyGroupId);
        }
    }

    public static void deletePublicKey(final CloudFrontClient cloudFrontClient,
final String publicKeyId) {
        GetPublicKeyResponse getResponse = cloudFrontClient.getPublicKey(b ->
b.id(publicKeyId));

        DeletePublicKeyResponse deleteResponse =
cloudFrontClient.deletePublicKey(builder -> builder
            .id(publicKeyId)
            .ifMatch(getResponse.eTag()));

        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Public Key [{}]", publicKeyId);
        }
    }
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [DeleteKeyGroup](#)
  - [DeleteOriginAccessControl](#)
  - [DeletePublicKey](#)

Eine Verteilung aktualisieren

Das folgende Codebeispiel zeigt, wie Sie eine Amazon- CloudFront Verteilung aktualisieren.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import software.amazon.awssdk.services.cloudfront.model.UpdateDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ModifyDistribution {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <id>\s

                Where:
                id - the id value of the distribution.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String id = args[0];
```



```
CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
    .region(Region.AWS_GLOBAL)
    .build();

modDistribution(cloudFrontClient, id);
cloudFrontClient.close();
}

public static void modDistribution(CloudFrontClient cloudFrontClient, String
idVal) {
    try {
        // Get the Distribution to modify.
        GetDistributionRequest disRequest = GetDistributionRequest.builder()
            .id(idVal)
            .build();

        GetDistributionResponse response =
cloudFrontClient.getDistribution(disRequest);
        Distribution disObject = response.distribution();
        DistributionConfig config = disObject.distributionConfig();

        // Create a new DistributionConfig object and add new values to comment
and
        // aliases
        DistributionConfig config1 = DistributionConfig.builder()
            .aliases(config.aliases()) // You can pass in new values here
            .comment("New Comment")
            .cacheBehaviors(config.cacheBehaviors())
            .priceClass(config.priceClass())
            .defaultCacheBehavior(config.defaultCacheBehavior())
            .enabled(config.enabled())
            .callerReference(config.callerReference())
            .logging(config.logging())
            .originGroups(config.originGroups())
            .origins(config.origins())
            .restrictions(config.restrictions())
            .defaultRootObject(config.defaultRootObject())
            .webACLId(config.webACLId())
            .httpVersion(config.httpVersion())
            .viewerCertificate(config.viewerCertificate())
            .customErrorResponses(config.customErrorResponses())
            .build();
```

```

        UpdateDistributionRequest updateDistributionRequest =
UpdateDistributionRequest.builder()
    .distributionConfig(config1)
    .id(disObject.id())
    .ifMatch(response.eTag())
    .build();

        cloudFrontClient.updateDistribution(updateDistributionRequest);

    } catch (CloudFrontException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

- Weitere API-Informationen finden Sie unter [UpdateDistribution](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Hochladen eines öffentlichen Schlüssels

Das folgende Codebeispiel zeigt, wie Sie einen öffentlichen Schlüssel hochladen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Das folgende Codebeispiel liest einen öffentlichen Schlüssel und lädt ihn in Amazon hoch CloudFront.

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreatePublicKeyResponse;
import software.amazon.awssdk.utils.IoUtils;

```

```
import java.io.IOException;
import java.io.InputStream;
import java.util.UUID;

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

    public static String createPublicKey(CloudFrontClient cloudFrontClient, String
        publicKeyFileName) {
        try (InputStream is =
            CreatePublicKey.class.getClassLoader().getResourceAsStream(publicKeyFileName)) {
            String publicKeyString = IoUtils.toUtf8String(is);
            CreatePublicKeyResponse createPublicKeyResponse = cloudFrontClient
                .createPublicKey(b -> b.publicKeyConfig(c -> c
                    .name("JavaCreatedPublicKey" + UUID.randomUUID())
                    .encodedKey(publicKeyString)
                    .callerReference(UUID.randomUUID().toString())));
            String createdPublicKeyId = createPublicKeyResponse.publicKey().id();
            logger.info("Public key created with id: [{}]", createdPublicKeyId);
            return createdPublicKeyId;

        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```


- Weitere API-Informationen finden Sie unter [CreatePublicKey](#) in der APIAWS SDK for Java 2.x - Referenz zu .

## Szenarien

### Signieren von URLs und Cookies

Das folgende Codebeispiel zeigt, wie signierte URLs und Cookies erstellt werden, die den Zugriff auf eingeschränkte Ressourcen ermöglichen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Verwenden Sie die [CannedSignerRequest](#) Klasse , um URLs oder Cookies mit einer vordefinierten Richtlinie zu signieren.

```
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCannedPolicyRequest {

    public static CannedSignerRequest createRequestForCannedPolicy(String
distributionDomainName,
        String fileNameToUpload,
        String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
resourcePath).toString();
        Instant expirationDate = Instant.now().plus(7, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CannedSignerRequest.builder()
            .resourceUrl(cloudFrontUrl)
            .privateKey(path)
            .keyPairId(publicKeyId)
            .expirationDate(expirationDate)
            .build();
    }
}
```

Verwenden Sie die [CustomSignerRequest](#) Klasse , um URLs oder Cookies mit einer benutzerdefinierten Richtlinie zu signieren. und activeDate ipRange sind optionale Methoden.

```
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCustomPolicyRequest {

    public static CustomSignerRequest createRequestForCustomPolicy(String
distributionDomainName,
        String fileNameToUpload,
        String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
resourcePath).toString();
        Instant expireDate = Instant.now().plus(7, ChronoUnit.DAYS);
        // URL will be accessible tomorrow using the signed URL.
        Instant activeDate = Instant.now().plus(1, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CustomSignerRequest.builder()
            .resourceUrl(cloudFrontUrl)
            .privateKey(path)
            .keyPairId(publicKeyId)
            .expirationDate(expireDate)
            .activeDate(activeDate) // Optional.
            // .ipRange("192.168.0.1/24") // Optional.
            .build();
    }
}
```

Das folgende Beispiel zeigt die Verwendung der [CloudFrontUtilities](#)Klasse zum Erstellen signierter Cookies und URLs. [Sehen Sie](#) sich dieses Codebeispiel auf an GitHub.

```
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontUtilities;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCannedPolicy;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCustomPolicy;
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;
import software.amazon.awssdk.services.cloudfront.url.SignedUrl;

public class SigningUtilities {
    private static final Logger logger =
    LoggerFactory.getLogger(SigningUtilities.class);
    private static final CloudFrontUtilities cloudFrontUtilities =
    CloudFrontUtilities.create();

    public static SignedUrl signUrlForCannedPolicy(CannedSignerRequest
    cannedSignerRequest) {
        SignedUrl signedUrl =
    cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static SignedUrl signUrlForCustomPolicy(CustomSignerRequest
    customSignerRequest) {
        SignedUrl signedUrl =
    cloudFrontUtilities.getSignedUrlWithCustomPolicy(customSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static CookiesForCannedPolicy
    getCookiesForCannedPolicy(CannedSignerRequest cannedSignerRequest) {
        CookiesForCannedPolicy cookiesForCannedPolicy = cloudFrontUtilities
            .getCookiesForCannedPolicy(cannedSignerRequest);
        logger.info("Cookie EXPIRES header [{}]",
    cookiesForCannedPolicy.expiresHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
    cookiesForCannedPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
    cookiesForCannedPolicy.signatureHeaderValue());
        return cookiesForCannedPolicy;
    }
}
```

```
public static CookiesForCustomPolicy
getCookiesForCustomPolicy(CustomSignerRequest customSignerRequest) {
    CookiesForCustomPolicy cookiesForCustomPolicy = cloudFrontUtilities
        .getCookiesForCustomPolicy(customSignerRequest);
    logger.info("Cookie POLICY header [{}]",
cookiesForCustomPolicy.policyHeaderValue());
    logger.info("Cookie KEYPAIR header [{}]",
cookiesForCustomPolicy.keyPairIdHeaderValue());
    logger.info("Cookie SIGNATURE header [{}]",
cookiesForCustomPolicy.signatureHeaderValue());
    return cookiesForCustomPolicy;
}
}
```

- Weitere API-Informationen finden Sie unter [CloudFrontUtilities](#) in der APIAWS SDK for Java 2.x -Referenz für .

## CloudWatch -Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit verwenden CloudWatch.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Erste Schritte

#### Hallo CloudWatch

Die folgenden Codebeispiele veranschaulichen, wie Sie mit der Verwendung von CloudWatch beginnen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloService {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <namespace>\s

                Where:
                namespace - The namespace to filter against (for example, AWS/
                EC2).\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String namespace = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
```



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

    listMets(cw, namespace);
    cw.close();
}

public static void listMets(CloudWatchClient cw, String namespace) {
    try {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsIterable listRes = cw.listMetricsPaginator(request);
        listRes.stream()
            .flatMap(r -> r.metrics().stream())
            .forEach(metrics -> System.out.println(" Retrieved metric is: "
+ metrics.metricName()));

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [ListMetrics](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Themen

- [Aktionen](#)
- [Szenarien](#)

## Aktionen

### Erstellen eines Dashboards

Das folgende Codebeispiel zeigt, wie Sie ein Amazon- CloudWatch Dashboard erstellen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {
    try {
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(readFileAsString(fileName))
            .build();

        PutDashboardResponse response = cw.putDashboard(dashboardRequest);
        System.out.println(dashboardName + " was successfully created.");
        List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
        if (messages.isEmpty()) {
            System.out.println("There are no messages in the new Dashboard");
        } else {
            for (DashboardValidationMessage message : messages) {
                System.out.println("Message is: " + message.message());
            }
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [PutDashboard](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Metrik-Alarm erstellen

Das folgende Codebeispiel zeigt, wie Sie einen Amazon- CloudWatch Alarm erstellen oder aktualisieren und ihn der angegebenen Metrik, dem mathematischen Metrikausdruck, dem Anomalieerkennungmodell oder der Metrics-Insights-Abfrage zuordnen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        String alarmName = rootNode.findValue("exampleAlarmName").asText();
        String emailTopic = rootNode.findValue("emailTopic").asText();
        String accountId = rootNode.findValue("accountId").asText();
        String region = rootNode.findValue("region").asText();

        // Create a List for alarm actions.
        List<String> alarmActions = new ArrayList<>();
        alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);
        PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
            .alarmActions(alarmActions)
            .alarmDescription("Example metric alarm")
            .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
            .threshold(100.00)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
```

```
        .evaluationPeriods(1)
        .period(10)
        .statistic("Maximum")
        .datapointsToAlarm(1)
        .treatMissingData("ignore")
        .build();

        cw.putMetricAlarm(alarmRequest);
        System.out.println(alarmName + " was successfully created!");
        return alarmName;

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- Weitere API-Informationen finden Sie unter [PutMetricAlarm](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Anomaliedetektor erstellen

Das folgende Codebeispiel zeigt, wie Sie einen Amazon CloudWatch -Anomaliedetektor erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
```

```
String customMetricName =
rootNode.findValue("customMetricName").asText();

SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
    .metricName(customMetricName)
    .namespace(customMetricNamespace)
    .stat("Maximum")
    .build();

PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
    .build();

cw.putAnomalyDetector(anomalyDetectorRequest);
System.out.println("Added anomaly detector for metric " +
customMetricName + ".");

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [PutAnomalyDetector](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Löschen von Alarmen

Das folgende Codebeispiel zeigt, wie Sie Amazon- CloudWatch Alarme löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteAlarm {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <alarmName>

            Where:
            alarmName - An alarm name to delete (for example, MyAlarm).
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarmName = args[0];
        Region region = Region.US_EAST_2;
        CloudWatchClient cw = CloudWatchClient.builder()
            .region(region)
            .build();

        deleteCWAlarm(cw, alarmName);
        cw.close();
    }

    public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
        try {
            DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
                .alarmNames(alarmName)
                .build();
        }
    }
}
```

```

        cw.deleteAlarms(request);
        System.out.printf("Successfully deleted alarm %s", alarmName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Weitere API-Informationen finden Sie unter [DeleteAlarms](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Anomaliedetektor löschen

Das folgende Codebeispiel zeigt, wie Sie einen Amazon CloudWatch -Anomaliedetektor löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

public static void deleteAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
            .metricName(customMetricName)

```

```
        .namespace(customMetricNamespace)
        .stat("Maximum")
        .build();

    DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
        .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
        .build();

    cw.deleteAnomalyDetector(request);
    System.out.println("Successfully deleted the Anomaly Detector.");

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
} catch (IOException e) {
    e.printStackTrace();
}
}
```

- Weitere API-Informationen finden Sie unter [DeleteAnomalyDetector](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Dashboards löschen

Das folgende Codebeispiel zeigt, wie Sie Amazon- CloudWatch Dashboards löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void deleteDashboard(CloudWatchClient cw, String dashboardName) {
    try {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
```



```
        .build();
        cw.deleteDashboards(dashboardsRequest);
        System.out.println("dashboardName + " was successfully deleted.");

    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteDashboards](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Beschreiben eines Alarmverlaufs

Das folgende Codebeispiel zeigt, wie Sie einen Amazon- CloudWatch Alarmverlauf beschreiben.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void getAlarmHistory(CloudWatchClient cw, String fileName, String
date) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String alarmName = rootNode.findValue("exampleAlarmName").asText();

        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();
        DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
            .startDate(start)
            .endDate(endDate)
```

```
        .alarmName(alarmName)
        .historyItemType(HistoryItemType.ACTION)
        .build();

    DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
    List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
    if (historyItems.isEmpty()) {
        System.out.println("No alarm history data found for " + alarmName +
".");
    } else {
        for (AlarmHistoryItem item : historyItems) {
            System.out.println("History summary: " + item.historySummary());
            System.out.println("Time stamp: " + item.timestamp());
        }
    }

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [DescribeAlarmHistory](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Alarmer beschreiben

Das folgende Codebeispiel zeigt, wie Sie Amazon- CloudWatch Alarmer beschreiben.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void describeAlarms(CloudWatchClient cw) {
    try {
```

```
List<AlarmType> typeList = new ArrayList<>();
typeList.add(AlarmType.METRIC_ALARM);

DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
    .alarmTypes(typeList)
    .maxRecords(10)
    .build();

DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
List<MetricAlarm> alarmList = response.metricAlarms();
for (MetricAlarm alarm : alarmList) {
    System.out.println("Alarm name: " + alarm.alarmName());
    System.out.println("Alarm description: " +
alarm.alarmDescription());
}
} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [DescribeAlarms](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beschreiben von Alarmen für eine Metrik

Das folgende Codebeispiel zeigt, wie Sie Amazon- CloudWatch Alarme für eine Metrik beschreiben.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void checkForMetricAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
```

```

        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        boolean hasAlarm = false;
        int retries = 10;

        DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        while (!hasAlarm && retries > 0) {
            DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
            hasAlarm = response.hasMetricAlarms();
            retries--;
            Thread.sleep(20000);
            System.out.println(".");
        }
        if (!hasAlarm)
            System.out.println("No Alarm state found for " + customMetricName +
" after 10 retries.");
        else
            System.out.println("Alarm state found for " + customMetricName +
".");

    } catch (CloudWatchException | IOException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}


```

- Weitere API-Informationen finden Sie unter [DescribeAlarmsForMetric](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Anomaliedetektoren beschreiben

Das folgende Codebeispiel zeigt, wie Sie Amazon CloudWatch -Anomaliedetektoren beschreiben.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void describeAnomalyDetectors(CloudWatchClient cw, String
fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
            .maxResults(10)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        DescribeAnomalyDetectorsResponse response =
cw.describeAnomalyDetectors(detectorsRequest);
        List<AnomalyDetector> anomalyDetectorList = response.anomalyDetectors();
        for (AnomalyDetector detector : anomalyDetectorList) {
            System.out.println("Metric name: " +
detector.singleMetricAnomalyDetector().metricName());
            System.out.println("State: " + detector.stateValue());
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeAnomalyDetectors](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Deaktivieren von Alarmaktionen

Das folgende Codebeispiel zeigt, wie Sie Amazon- CloudWatch Alarmaktionen deaktivieren.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DisableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DisableAlarmActions {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <alarmName>

                Where:
                alarmName - An alarm name to disable (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String alarmName = args[0];
    Region region = Region.US_EAST_1;
    CloudWatchClient cw = CloudWatchClient.builder()
        .region(region)
        .build();

    disableActions(cw, alarmName);
    cw.close();
}

public static void disableActions(CloudWatchClient cw, String alarmName) {
    try {
        DisableAlarmActionsRequest request =
DisableAlarmActionsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.disableAlarmActions(request);
        System.out.printf("Successfully disabled actions on alarm %s",
alarmName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [DisableAlarmActions](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Aktivieren von Alarmaktionen

Das folgende Codebeispiel zeigt, wie Sie Amazon- CloudWatch Alarmaktionen aktivieren.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.EnableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EnableAlarmActions {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <alarmName>

                Where:
                alarmName - An alarm name to enable (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarm = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();
```



```
        enableActions(cw, alarm);
        cw.close();
    }

    public static void enableActions(CloudWatchClient cw, String alarm) {
        try {
            EnableAlarmActionsRequest request = EnableAlarmActionsRequest.builder()
                .alarmNames(alarm)
                .build();

            cw.enableAlarmActions(request);
            System.out.printf("Successfully enabled actions on alarm %s", alarm);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [EnableAlarmActions](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Metrikdaten-Image abrufen

Das folgende Codebeispiel zeigt, wie Sie ein Amazon- CloudWatch Metrikdaten-Image abrufen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void getAndOpenMetricImage(CloudWatchClient cw, String fileName) {
    System.out.println("Getting Image data for custom metric.");
    try {
        String myJSON = "{\n" +
```

```

"  \"title\": \"Example Metric Graph\", \"\n\" +
"  \"view\": \"timeSeries\", \"\n\" +
"  \"stacked\": false, \"\n\" +
"  \"period\": 10, \"\n\" +
"  \"width\": 1400, \"\n\" +
"  \"height\": 600, \"\n\" +
"  \"metrics\": [ \"\n\" +
"    [ \"\n\" +
"      \"AWS/Billing\", \"\n\" +
"      \"EstimatedCharges\", \"\n\" +
"      \"Currency\", \"\n\" +
"      \"USD\" \"\n\" +
"    ] \"\n\" +
"  ] \"\n\" +
"}";

```

```

    GetMetricWidgetImageRequest imageRequest =
    GetMetricWidgetImageRequest.builder()
        .metricWidget(myJSON)
        .build();

    GetMetricWidgetImageResponse response =
    cw.getMetricWidgetImage(imageRequest);
    SdkBytes sdkBytes = response.metricWidgetImage();
    byte[] bytes = sdkBytes.asByteArray();
    File outputFile = new File(fileName);
    try (FileOutputStream outputStream = new FileOutputStream(outputFile)) {
        outputStream.write(bytes);
    }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

```

- Weitere API-Informationen finden Sie unter [GetMetricWidgetImage](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Metrikdaten abrufen

Das folgende Codebeispiel zeigt, wie Sie Amazon- CloudWatch Metrikdaten abrufen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void getCustomMetricData(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set the date.
        Instant nowDate = Instant.now();

        long hours = 1;
        long minutes = 30;
        Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
ChronoUnit.MINUTES);

        Metric met = Metric.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        MetricStat metStat = MetricStat.builder()
            .stat("Maximum")
            .period(1)
            .metric(met)
            .build();

        MetricDataQuery dataQuery = MetricDataQuery.builder()
            .metricStat(metStat)
            .id("foo2")
            .returnData(true)
```

```
        .build();

List<MetricDataQuery> dq = new ArrayList<>();
dq.add(dataQuery);

GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
    .maxDatapoints(10)
    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
    .startTime(nowDate)
    .endTime(date2)
    .metricDataQueries(dq)
    .build();

GetMetricDataResponse response = cw.getMetricData(getMetReq);
List<MetricDataResult> data = response.metricDataResults();
for (MetricDataResult item : data) {
    System.out.println("The label is " + item.label());
    System.out.println("The status code is " +
item.statusCode().toString());
}

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [GetMetricData](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Abrufen von Metrikstatistiken

Das folgende Codebeispiel zeigt, wie Sie Amazon- CloudWatch Metrikstatistiken abrufen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
nameSpace, String metVal,
        String metricOption, String date, Dimension myDimension) {
    try {
        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();

        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
        .endTime(endDate)
        .startTime(start)
        .dimensions(myDimension)
        .metricName(metVal)
        .namespace(nameSpace)
        .period(86400)
        .statistics(Statistic.fromValue(metricOption))
        .build();

        GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
        List<Datapoint> data = response.datapoints();
        if (!data.isEmpty()) {
            for (Datapoint datapoint : data) {
                System.out
                    .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
            }
        } else {
            System.out.println("The returned data list is empty");
        }

    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [GetMetricStatistics](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Auflisten von Dashboards

Das folgende Codebeispiel zeigt, wie Sie Amazon- CloudWatch Dashboards auflisten.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void listDashboards(CloudWatchClient cw) {
    try {
        ListDashboardsIterable listRes = cw.listDashboardsPaginator();
        listRes.stream()
            .flatMap(r -> r.dashboardEntries().stream())
            .forEach(entry -> {
                System.out.println("Dashboard name is: " +
entry.dashboardName());
                System.out.println("Dashboard ARN is: " +
entry.dashboardArn());
            });

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [ListDashboards](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Metriken

Das folgende Codebeispiel zeigt, wie Sie die Metadaten für Amazon- CloudWatch Metriken auflisten. Um Daten für eine Metrik abzurufen, verwenden Sie die GetMetricStatistics Aktionen GetMetricData oder .

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListMetrics {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <namespace>\s

                Where:
                namespace - The namespace to filter against (for example, AWS/
EC2).\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String namespace = args[0];
        Region region = Region.US_EAST_1;
```

```
CloudWatchClient cw = CloudWatchClient.builder()
    .region(region)
    .build();

listMets(cw, namespace);
cw.close();
}

public static void listMets(CloudWatchClient cw, String namespace) {
    boolean done = false;
    String nextToken = null;

    try {
        while (!done) {

            ListMetricsResponse response;
            if (nextToken == null) {
                ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

                response = cw.listMetrics(request);
            } else {
                ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .nextToken(nextToken)
                    .build();

                response = cw.listMetrics(request);
            }

            for (Metric metric : response.metrics()) {
                System.out.printf("Retrieved metric %s", metric.metricName());
                System.out.println();
            }

            if (response.nextToken() == null) {
                done = true;
            } else {
                nextToken = response.nextToken();
            }
        }
    } catch (CloudWatchException e) {
```



```

        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Weitere API-Informationen finden Sie unter [ListMetrics](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Einfügen von Daten in eine Metrik

Das folgende Codebeispiel zeigt, wie Metrikdatenpunkte in Amazon veröffentlicht werden CloudWatch.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

public static void addMetricDataForAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set an Instant object.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName(customMetricName)

```

```

        .unit(StandardUnit.NONE)
        .value(1001.00)
        .timestamp(instant)
        .build();

    MetricDatum datum2 = MetricDatum.builder()
        .metricName(customMetricName)
        .unit(StandardUnit.NONE)
        .value(1002.00)
        .timestamp(instant)
        .build();

    List<MetricDatum> metricDataList = new ArrayList<>();
    metricDataList.add(datum);
    metricDataList.add(datum2);

    PutMetricDataRequest request = PutMetricDataRequest.builder()
        .namespace(customMetricNamespace)
        .metricData(metricDataList)
        .build();

    cw.putMetricData(request);
    System.out.println("Added metric values for for metric " +
customMetricName);

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

```

- Weitere API-Informationen finden Sie unter [PutMetricData](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Szenarien

Erste Schritte mit CloudWatch-Metriken, -Dashboards und -Alarmen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Auflisten von CloudWatch Namespaces und Metriken.

- 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 Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.AlarmHistoryItem;
import software.amazon.awssdk.services.cloudwatch.model.AlarmType;
import software.amazon.awssdk.services.cloudwatch.model.AnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ComparisonOperator;
import software.amazon.awssdk.services.cloudwatch.model.DashboardValidationMessage;
import software.amazon.awssdk.services.cloudwatch.model.Datapoint;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DeleteAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.DeleteDashboardsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryResponse;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricRequest;
```

```
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricResponse;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsResponse;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataResponse;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsRequest;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsResponse;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageResponse;
import software.amazon.awssdk.services.cloudwatch.model.HistoryItemType;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;
import software.amazon.awssdk.services.cloudwatch.model.MetricAlarm;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataQuery;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataResult;
import software.amazon.awssdk.services.cloudwatch.model.MetricDatum;
import software.amazon.awssdk.services.cloudwatch.model.MetricStat;
import software.amazon.awssdk.services.cloudwatch.model.PutAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardResponse;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricAlarmRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.ScanBy;
import software.amazon.awssdk.services.cloudwatch.model.SingleMetricAnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.Statistic;
import software.amazon.awssdk.services.cloudwatch.paginators.ListDashboardsIterable;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.time.Instant;
```

```
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To enable billing metrics and statistics for this example, make sure billing
 * alerts are enabled for your account:
 * https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/monitor\_estimated\_charges\_with\_cloudwatch.html#turning\_on\_billing\_metrics
 *
 * This Java code example performs the following tasks:
 *
 * 1. List available namespaces from Amazon CloudWatch.
 * 2. List available metrics within the selected Namespace.
 * 3. Get statistics for the selected metric over the last day.
 * 4. Get CloudWatch estimated billing for the last week.
 * 5. Create a new CloudWatch dashboard with metrics.
 * 6. List dashboards using a paginator.
 * 7. Create a new custom metric by adding data for it.
 * 8. Add the custom metric to the dashboard.
 * 9. Create an alarm for the custom metric.
 * 10. Describe current alarms.
 * 11. Get current data for the new custom metric.
 * 12. Push data into the custom metric to trigger the alarm.
 * 13. Check the alarm state using the action DescribeAlarmsForMetric.
 * 14. Get alarm history for the new alarm.
 * 15. Add an anomaly detector for the custom metric.
 * 16. Describe current anomaly detectors.
 * 17. Get a metric image for the custom metric.
 * 18. Clean up the Amazon CloudWatch resources.
 */
public class CloudWatchScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
```

```
public static void main(String[] args) throws IOException {
    final String usage = ""

        Usage:
            <myDate> <costDateWeek> <dashboardName> <dashboardJson>
<dashboardAdd> <settings> <metricImage> \s

        Where:
            myDate - The start date to use to get metric statistics. (For
example, 2023-01-11T18:35:24.00Z.)\s
            costDateWeek - The start date to use to get AWS/Billinget
statistics. (For example, 2023-01-11T18:35:24.00Z.)\s
            dashboardName - The name of the dashboard to create.\s
            dashboardJson - The location of a JSON file to use to create a
dashboard. (See Readme file.)\s
            dashboardAdd - The location of a JSON file to use to update a
dashboard. (See Readme file.)\s
            settings - The location of a JSON file from which various values
are read. (See Readme file.)\s
            metricImage - The location of a BMP file that is used to create a
graph.\s

        """;

    if (args.length != 7) {
        System.out.println(usage);
        System.exit(1);
    }

    Region region = Region.US_EAST_1;
    String myDate = args[0];
    String costDateWeek = args[1];
    String dashboardName = args[2];
    String dashboardJson = args[3];
    String dashboardAdd = args[4];
    String settings = args[5];
    String metricImage = args[6];

    Double dataPoint = Double.parseDouble("10.0");
    Scanner sc = new Scanner(System.in);
    CloudWatchClient cw = CloudWatchClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();
```

```
System.out.println(DASHES);
System.out.println("Welcome to the Amazon CloudWatch example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(
    "1. List at least five available unique namespaces from Amazon
CloudWatch. Select one from the list.");
ArrayList<String> list = listNameSpaces(cw);
for (int z = 0; z < 5; z++) {
    int index = z + 1;
    System.out.println("    " + index + ". " + list.get(z));
}

String selectedNamespace = "";
String selectedMetrics = "";
int num = Integer.parseInt(sc.nextLine());
if (1 <= num && num <= 5) {
    selectedNamespace = list.get(num - 1);
} else {
    System.out.println("You did not select a valid option.");
    System.exit(1);
}
System.out.println("You selected " + selectedNamespace);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. List available metrics within the selected namespace
and select one from the list.");
ArrayList<String> metList = listMets(cw, selectedNamespace);
for (int z = 0; z < 5; z++) {
    int index = z + 1;
    System.out.println("    " + index + ". " + metList.get(z));
}
num = Integer.parseInt(sc.nextLine());
if (1 <= num && num <= 5) {
    selectedMetrics = metList.get(num - 1);
} else {
    System.out.println("You did not select a valid option.");
    System.exit(1);
}
System.out.println("You selected " + selectedMetrics);
Dimension myDimension = getSpecificMet(cw, selectedNamespace);
```

```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Get statistics for the selected metric over the last
day.");
String metricOption = "";
ArrayList<String> statTypes = new ArrayList<>();
statTypes.add("SampleCount");
statTypes.add("Average");
statTypes.add("Sum");
statTypes.add("Minimum");
statTypes.add("Maximum");

for (int t = 0; t < 5; t++) {
    System.out.println("    " + (t + 1) + ". " + statTypes.get(t));
}
System.out.println("Select a metric statistic by entering a number from the
preceding list:");
num = Integer.parseInt(sc.nextLine());
if (1 <= num && num <= 5) {
    metricOption = statTypes.get(num - 1);
} else {
    System.out.println("You did not select a valid option.");
    System.exit(1);
}
System.out.println("You selected " + metricOption);
getAndDisplayMetricStatistics(cw, selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Get CloudWatch estimated billing for the last
week.");
getMetricStatistics(cw, costDateWeek);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Create a new CloudWatch dashboard with metrics.");
createDashboardWithMetrics(cw, dashboardName, dashboardJson);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. List dashboards using a paginator.");
listDashboards(cw);
```



```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Create a new custom metric by adding data to it.");
createNewCustomMetric(cw, dataPoint);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Add an additional metric to the dashboard.");
addMetricToDashboard(cw, dashboardAdd, dashboardName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Create an alarm for the custom metric.");
String alarmName = createAlarm(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Describe ten current alarms.");
describeAlarms(cw);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Get current data for new custom metric.");
getCustomMetricData(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Push data into the custom metric to trigger the
alarm.");
addMetricDataForAlarm(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Check the alarm state using the action
DescribeAlarmsForMetric.");
checkForMetricAlarm(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Get alarm history for the new alarm.");
getAlarmHistory(cw, settings, myDate);
System.out.println(DASHES);
```

```

        System.out.println(DASHES);
        System.out.println("15. Add an anomaly detector for the custom metric.");
        addAnomalyDetector(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Describe current anomaly detectors.");
        describeAnomalyDetectors(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Get a metric image for the custom metric.");
        getAndOpenMetricImage(cw, metricImage);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Clean up the Amazon CloudWatch resources.");
        deleteDashboard(cw, dashboardName);
        deleteCWAlarm(cw, alarmName);
        deleteAnomalyDetector(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Amazon CloudWatch example scenario is complete.");
        System.out.println(DASHES);
        cw.close();
    }

    public static void deleteAnomalyDetector(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .stat("Maximum")

```

```
        .build();

        DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
        .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
        .build();

        cw.deleteAnomalyDetector(request);
        System.out.println("Successfully deleted the Anomaly Detector.");

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
    try {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.deleteAlarms(request);
        System.out.println("Successfully deleted alarm " + alarmName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteDashboard(CloudWatchClient cw, String dashboardName) {
    try {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();

        cw.deleteDashboards(dashboardsRequest);
        System.out.println(dashboardName + " was successfully deleted.");

    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
    }
}
```

```
        System.exit(1);
    }
}

public static void getAndOpenMetricImage(CloudWatchClient cw, String fileName) {
    System.out.println("Getting Image data for custom metric.");
    try {
        String myJSON = "{\n" +
            "  \"title\": \"Example Metric Graph\",\n" +
            "  \"view\": \"timeSeries\",\n" +
            "  \"stacked\": false,\n" +
            "  \"period\": 10,\n" +
            "  \"width\": 1400,\n" +
            "  \"height\": 600,\n" +
            "  \"metrics\": [\n" +
            "    [\n" +
            "      \"AWS/Billing\",\n" +
            "      \"EstimatedCharges\",\n" +
            "      \"Currency\",\n" +
            "      \"USD\"\n" +
            "    ]\n" +
            "  ]\n" +
            "}";

        GetMetricWidgetImageRequest imageRequest =
            GetMetricWidgetImageRequest.builder()
                .metricWidget(myJSON)
                .build();

        GetMetricWidgetImageResponse response =
            cw.getMetricWidgetImage(imageRequest);
        SdkBytes sdkBytes = response.metricWidgetImage();
        byte[] bytes = sdkBytes.asByteArray();
        File outputFile = new File(fileName);
        try (FileOutputStream outputStream = new FileOutputStream(outputFile)) {
            outputStream.write(bytes);
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
public static void describeAnomalyDetectors(CloudWatchClient cw, String
fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
            .maxResults(10)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        DescribeAnomalyDetectorsResponse response =
cw.describeAnomalyDetectors(detectorsRequest);
        List<AnomalyDetector> anomalyDetectorList = response.anomalyDetectors();
        for (AnomalyDetector detector : anomalyDetectorList) {
            System.out.println("Metric name: " +
detector.singleMetricAnomalyDetector().metricName());
            System.out.println("State: " + detector.stateValue());
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
```

```
        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
    .metricName(customMetricName)
    .namespace(customMetricNamespace)
    .stat("Maximum")
    .build();

        PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
    .build();

        cw.putAnomalyDetector(anomalyDetectorRequest);
        System.out.println("Added anomaly detector for metric " +
customMetricName + ".");

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

    public static void getAlarmHistory(CloudWatchClient cw, String fileName, String
date) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String alarmName = rootNode.findValue("exampleAlarmName").asText();

            Instant start = Instant.parse(date);
            Instant endDate = Instant.now();
            DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                .startDate(start)
                .endDate(endDate)
                .alarmName(alarmName)
                .historyItemType(HistoryItemType.ACTION)
                .build();

            DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
            List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
```

```
        if (historyItems.isEmpty()) {
            System.out.println("No alarm history data found for " + alarmName +
                ".");
        } else {
            for (AlarmHistoryItem item : historyItems) {
                System.out.println("History summary: " + item.historySummary());
                System.out.println("Time stamp: " + item.timestamp());
            }
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void checkForMetricAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        boolean hasAlarm = false;
        int retries = 10;

        DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        while (!hasAlarm && retries > 0) {
            DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
            hasAlarm = response.hasMetricAlarms();
            retries--;
            Thread.sleep(20000);
            System.out.println(".");
        }
        if (!hasAlarm)
```

```
        System.out.println("No Alarm state found for " + customMetricName +
" after 10 retries.");
    } else
        System.out.println("Alarm state found for " + customMetricName +
".");

    } catch (CloudWatchException | IOException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void addMetricDataForAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set an Instant object.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1001.00)
            .timestamp(instant)
            .build();

        MetricDatum datum2 = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1002.00)
            .timestamp(instant)
            .build();

        List<MetricDatum> metricDataList = new ArrayList<>();
        metricDataList.add(datum);
    }
}
```



```
metricDataList.add(datum2);

PutMetricDataRequest request = PutMetricDataRequest.builder()
    .namespace(customMetricNamespace)
    .metricData(metricDataList)
    .build();

cw.putMetricData(request);
System.out.println("Added metric values for for metric " +
customMetricName);

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getCustomMetricData(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set the date.
        Instant nowDate = Instant.now();

        long hours = 1;
        long minutes = 30;
        Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
ChronoUnit.MINUTES);

        Metric met = Metric.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        MetricStat metStat = MetricStat.builder()
            .stat("Maximum")
            .period(1)
```

```
        .metric(met)
        .build();

    MetricDataQuery dataQuery = MetricDataQuery.builder()
        .metricStat(metStat)
        .id("foo2")
        .returnData(true)
        .build();

    List<MetricDataQuery> dq = new ArrayList<>();
    dq.add(dataQuery);

    GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
        .maxDatapoints(10)
        .scanBy(ScanBy.TIMESTAMP_DESCENDING)
        .startTime(nowDate)
        .endTime(date2)
        .metricDataQueries(dq)
        .build();

    GetMetricDataResponse response = cw.getMetricData(getMetReq);
    List<MetricDataResult> data = response.metricDataResults();
    for (MetricDataResult item : data) {
        System.out.println("The label is " + item.label());
        System.out.println("The status code is " +
item.statusCode().toString());
    }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void describeAlarms(CloudWatchClient cw) {
    try {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);

        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();
```

```
DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
List<MetricAlarm> alarmList = response.metricAlarms();
for (MetricAlarm alarm : alarmList) {
    System.out.println("Alarm name: " + alarm.alarmName());
    System.out.println("Alarm description: " +
alarm.alarmDescription());
}
} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static String createAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        String alarmName = rootNode.findValue("exampleAlarmName").asText();
        String emailTopic = rootNode.findValue("emailTopic").asText();
        String accountId = rootNode.findValue("accountId").asText();
        String region = rootNode.findValue("region").asText();

        // Create a List for alarm actions.
        List<String> alarmActions = new ArrayList<>();
        alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);
        PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
            .alarmActions(alarmActions)
            .alarmDescription("Example metric alarm")
            .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
            .threshold(100.00)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .evaluationPeriods(1)
            .period(10)
            .statistic("Maximum")
```

```
        .datapointsToAlarm(1)
        .treatMissingData("ignore")
        .build();

    cw.putMetricAlarm(alarmRequest);
    System.out.println(alarmName + " was successfully created!");
    return alarmName;

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}

public static void addMetricToDashboard(CloudWatchClient cw, String fileName,
String dashboardName) {
    try {
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(readFileAsString(fileName))
            .build();

        cw.putDashboard(dashboardRequest);
        System.out.println(dashboardName + " was successfully updated.");

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void createNewCustomMetric(CloudWatchClient cw, Double dataPoint)
{
    try {
        Dimension dimension = Dimension.builder()
            .name("UNIQUE_PAGES")
            .value("URLS")
            .build();

        // Set an Instant object.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);
```

```
        MetricDatum datum = MetricDatum.builder()
            .metricName("PAGES_VISITED")
            .unit(StandardUnit.NONE)
            .value(dataPoint)
            .timestamp(instant)
            .dimensions(dimension)
            .build();

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace("SITE/TRAFFIC")
            .metricData(datum)
            .build();

        cw.putMetricData(request);
        System.out.println("Added metric values for for metric PAGES_VISITED");

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void listDashboards(CloudWatchClient cw) {
    try {
        ListDashboardsIterable listRes = cw.listDashboardsPaginator();
        listRes.stream()
            .flatMap(r -> r.dashboardEntries().stream())
            .forEach(entry -> {
                System.out.println("Dashboard name is: " +
entry.dashboardName());
                System.out.println("Dashboard ARN is: " +
entry.dashboardArn());
            });

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {
        try {
```

```
PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
    .dashboardName(dashboardName)
    .dashboardBody(readFileAsString(fileName))
    .build();

PutDashboardResponse response = cw.putDashboard(dashboardRequest);
System.out.println(dashboardName + " was successfully created.");
List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
if (messages.isEmpty()) {
    System.out.println("There are no messages in the new Dashboard");
} else {
    for (DashboardValidationMessage message : messages) {
        System.out.println("Message is: " + message.message());
    }
}

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static String readFileAsString(String file) throws IOException {
    return new String(Files.readAllBytes(Paths.get(file)));
}

public static void getMetricStatistics(CloudWatchClient cw, String costDateWeek)
{
    try {
        Instant start = Instant.parse(costDateWeek);
        Instant endDate = Instant.now();
        Dimension dimension = Dimension.builder()
            .name("Currency")
            .value("USD")
            .build();

        List<Dimension> dimensionList = new ArrayList<>();
        dimensionList.add(dimension);
        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
            .metricName("EstimatedCharges")
            .namespace("AWS/Billing")
            .dimensions(dimensionList)
```

```
        .statistics(Statistic.MAXIMUM)
        .startTime(start)
        .endTime(endDate)
        .period(86400)
        .build();

    GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
    List<Datapoint> data = response.datapoints();
    if (!data.isEmpty()) {
        for (Datapoint datapoint : data) {
            System.out
                .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
        }
    } else {
        System.out.println("The returned data list is empty");
    }

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
nameSpace, String metVal,
String metricOption, String date, Dimension myDimension) {
    try {
        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();

        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
            .endTime(endDate)
            .startTime(start)
            .dimensions(myDimension)
            .metricName(metVal)
            .namespace(nameSpace)
            .period(86400)
            .statistics(Statistic.fromValue(metricOption))
            .build();
```

```
        GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
        List<Datapoint> data = response.datapoints();
        if (!data.isEmpty()) {
            for (Datapoint datapoint : data) {
                System.out
                    .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
            }
        } else {
            System.out.println("The returned data list is empty");
        }

    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static Dimension getSpecificMet(CloudWatchClient cw, String namespace) {
    try {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsResponse response = cw.listMetrics(request);
        List<Metric> myList = response.metrics();
        Metric metric = myList.get(0);
        return metric.dimensions().get(0);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static ArrayList<String> listMets(CloudWatchClient cw, String namespace)
{
    try {
        ArrayList<String> metList = new ArrayList<>();
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();
```



```

        ListMetricsIterable listRes = cw.listMetricsPaginator(request);
        listRes.stream()
            .flatMap(r -> r.metrics().stream())
            .forEach(metrics -> metList.add(metrics.metricName()));

        return metList;

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static ArrayList<String> listNameSpaces(CloudWatchClient cw) {
    try {
        ArrayList<String> nameSpaceList = new ArrayList<>();
        ListMetricsRequest request = ListMetricsRequest.builder()
            .build();

        ListMetricsIterable listRes = cw.listMetricsPaginator(request);
        listRes.stream()
            .flatMap(r -> r.metrics().stream())
            .forEach(metrics -> {
                String data = metrics.namespace();
                if (!nameSpaceList.contains(data)) {
                    nameSpaceList.add(data);
                }
            });

        return nameSpaceList;
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
}

```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
- [DeleteAlarms](#)

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

## CloudWatch Beispiele für Ereignisse mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit - CloudWatch Ereignissen verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Themen

- [Aktionen](#)

## Aktionen

### Hinzufügen eines Tags

Das folgende Codebeispiel zeigt, wie Sie einem Amazon CloudWatch Events-Ereignis ein Ziel hinzufügen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.Target;

/**
 * To run this Java V2 code example, ensure that you have setup your development
 * environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutTargets {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <ruleName> <functionArn> <targetId>\s

                Where:
                ruleName - A rule name (for example, myrule).
                functionArn - An AWS Lambda function ARN (for example,
                arn:aws:lambda:us-west-2:xxxxxx047983:function:lamda1).
                targetId - A target id value.
                "";
```

```
        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        String functionArn = args[1];
        String targetId = args[2];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
            .build();

        putCWTargets(cwe, ruleName, functionArn, targetId);
        cwe.close();
    }

    public static void putCWTargets(CloudWatchEventsClient cwe, String ruleName,
        String functionArn, String targetId) {
        try {
            Target target = Target.builder()
                .arn(functionArn)
                .id(targetId)
                .build();

            PutTargetsRequest request = PutTargetsRequest.builder()
                .targets(target)
                .rule(ruleName)
                .build();

            cwe.putTargets(request);
            System.out.printf(
                "Successfully created CloudWatch events target for rule %s",
                ruleName);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [PutTargets](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen einer geplanten Regel

Das folgende Codebeispiel zeigt, wie Sie eine geplante Regel für Amazon CloudWatch Events erstellen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.RuleState;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutRule {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <ruleName> roleArn\s

                Where:
                ruleName - A rule name (for example, myrule).
                roleArn - A role ARN value (for example,
                arn:aws:iam::xxxxxx047983:user/MyUser).
                """;

        if (args.length != 2) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String ruleName = args[0];
    String roleArn = args[1];
    CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
        .build();

    putCWRule(cwe, ruleName, roleArn);
    cwe.close();
}

public static void putCWRule(CloudWatchEventsClient cwe, String ruleName, String
roleArn) {
    try {
        PutRuleRequest request = PutRuleRequest.builder()
            .name(ruleName)
            .roleArn(roleArn)
            .scheduleExpression("rate(5 minutes)")
            .state(RuleState.ENABLED)
            .build();

        PutRuleResponse response = cwe.putRule(request);
        System.out.printf(
            "Successfully created CloudWatch events rule %s with arn %s",
            roleArn, response.ruleArn());

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [PutRule](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Senden von Ereignissen

Das folgende Codebeispiel zeigt, wie Sie Amazon CloudWatch Events-Ereignisse senden.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequestEntry;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutEvents {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <resourceArn>

                Where:
                resourceArn - An Amazon Resource Name (ARN) related to the
events.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String resourceArn = args[0];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();
```

```
        putCWEvents(cwe, resourceArn);
        cwe.close();
    }

    public static void putCWEvents(CloudWatchEventsClient cwe, String resourceArn) {
        try {
            final String EVENT_DETAILS = "{ \"key1\": \"value1\", \"key2\": \"value2\" }";

            PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
                .detail(EVENT_DETAILS)
                .detailType("sampleSubmitted")
                .resources(resourceArn)
                .source("aws-sdk-java-cloudwatch-example")
                .build();

            PutEventsRequest request = PutEventsRequest.builder()
                .entries(requestEntry)
                .build();

            cwe.putEvents(request);
            System.out.println("Successfully put CloudWatch event");

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [PutEvents](#) in der APIAWS SDK for Java 2.x - Referenz für .

## CloudWatch Protokolliert Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit - CloudWatch Protokollen verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.



Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)

## Aktionen

### Erstellen eines Abonnementfilters

Das folgende Codebeispiel zeigt, wie Sie einen Amazon- CloudWatch Logs-Abonnementfilter erstellen.

SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.PutSubscriptionFilterRequest;

/**
 * Before running this code example, you need to grant permission to CloudWatch
 * Logs the right to execute your Lambda function.
 * To perform this task, you can use this CLI command:
 *
 * aws lambda add-permission --function-name "lamda1" --statement-id "lamda1"
 * --principal "logs.us-west-2.amazonaws.com" --action "lambda:InvokeFunction"
 * --source-arn "arn:aws:logs:us-west-2:111111111111:log-group:testgroup:*"
 * --source-account "111111111111"
 *
 * Make sure you replace the function name with your function name and replace
```

```
* '111111111111' with your account details.  
* For more information, see "Subscription Filters with AWS Lambda" in the  
* Amazon CloudWatch Logs Guide.  
*  
*  
* Also, before running this Java V2 code example, set up your development  
* environment, including your credentials.  
*  
* For more information, see the following documentation topic:  
*  
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
*  
*/
```

```
public class PutSubscriptionFilter {  
    public static void main(String[] args) {  
        final String usage = ""  
  
            Usage:  
            <filter> <pattern> <logGroup> <functionArn>\s  
  
            Where:  
            filter - A filter name (for example, myfilter).  
            pattern - A filter pattern (for example, ERROR).  
            logGroup - A log group name (testgroup).  
            functionArn - An AWS Lambda function ARN (for example,  
arn:aws:lambda:us-west-2:111111111111:function:lambda1) .  
            "";  
  
        if (args.length != 4) {  
            System.out.println(usage);  
            System.exit(1);  
        }  
  
        String filter = args[0];  
        String pattern = args[1];  
        String logGroup = args[2];  
        String functionArn = args[3];  
        Region region = Region.US_WEST_2;  
        CloudWatchLogsClient cwl = CloudWatchLogsClient.builder()  
            .region(region)  
            .build();  
  
        putSubFilters(cwl, filter, pattern, logGroup, functionArn);  
    }  
}
```

```
        cwl.close();
    }

    public static void putSubFilters(CloudWatchLogsClient cwl,
        String filter,
        String pattern,
        String logGroup,
        String functionArn) {

        try {
            PutSubscriptionFilterRequest request =
                PutSubscriptionFilterRequest.builder()
                    .filterName(filter)
                    .filterPattern(pattern)
                    .logGroupName(logGroup)
                    .destinationArn(functionArn)
                    .build();

            cwl.putSubscriptionFilter(request);
            System.out.printf(
                "Successfully created CloudWatch logs subscription filter %s",
                filter);

        } catch (CloudWatchLogsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [PutSubscriptionFilter](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Löschen eines Abonnementfilters

Das folgende Codebeispiel zeigt, wie Sie einen Amazon- CloudWatch Logs-Abonnementfilter löschen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
  software.amazon.awssdk.services.cloudwatchlogs.model.DeleteSubscriptionFilterRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <filter> <logGroup>

                Where:
                filter - The name of the subscription filter (for example,
MyFilter).
                logGroup - The name of the log group. (for example, testgroup).
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String filter = args[0];
        String logGroup = args[1];
        CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
```

```
        .build();

        deleteSubFilter(logs, filter, logGroup);
        logs.close();
    }

    public static void deleteSubFilter(CloudWatchLogsClient logs, String filter,
String logGroup) {
        try {
            DeleteSubscriptionFilterRequest request =
DeleteSubscriptionFilterRequest.builder()
                .filterName(filter)
                .logGroupName(logGroup)
                .build();

            logs.deleteSubscriptionFilter(request);
            System.out.printf("Successfully deleted CloudWatch logs subscription
filter %s", filter);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteSubscriptionFilter](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Beschreiben von bestehenden Abonnementfiltern

Das folgende Codebeispiel zeigt, wie vorhandene Abonnementfilter von Amazon CloudWatch Logs beschrieben werden.

### SDK für Java 2.x

#### Note

Auf [GitHub](#) gibt es mehr Beispiele. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersRequest;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.SubscriptionFilter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeSubscriptionFilters {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
            <logGroup>

            Where:
            logGroup - A log group name (for example, myloggroup).
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String logGroup = args[0];
        CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        describeFilters(logs, logGroup);
        logs.close();
    }
}
```

```
public static void describeFilters(CloudWatchLogsClient logs, String logGroup) {
    try {
        boolean done = false;
        String newToken = null;

        while (!done) {
            DescribeSubscriptionFiltersResponse response;
            if (newToken == null) {
                DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                    .logGroupName(logGroup)
                    .limit(1).build();

                response = logs.describeSubscriptionFilters(request);
            } else {
                DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                    .nextToken(newToken)
                    .logGroupName(logGroup)
                    .limit(1).build();
                response = logs.describeSubscriptionFilters(request);
            }

            for (SubscriptionFilter filter : response.subscriptionFilters()) {
                System.out.printf("Retrieved filter with name %s, " + "pattern
%s " + "and destination arn %s",
                    filter.filterName(),
                    filter.filterPattern(),
                    filter.destinationArn());
            }

            if (response.nextToken() == null) {
                done = true;
            } else {
                newToken = response.nextToken();
            }
        }

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.printf("Done");
}
```

```
}

```

- Weitere API-Informationen finden Sie unter [DescribeSubscriptionFilters](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Starten einer Live-Tail-Sitzung

Das folgende Codebeispiel zeigt, wie Sie eine Live-Tail-Sitzung für eine vorhandene Protokollgruppe/ einen vorhandenen Protokollstream starten.

### SDK für Java 2.x

Binden Sie die erforderlichen Dateien ein.

```
import io.reactivex.FlowableSubscriber;
import io.reactivex.annotations.NonNull;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsAsyncClient;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionLogEvent;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionStart;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionUpdate;
import software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailRequest;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseHandler;
import software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseStream;

import java.util.Date;
import java.util.List;
import java.util.concurrent.atomic.AtomicReference;
```

Verarbeiten Sie die Ereignisse aus der Live-Tail-Sitzung.

```
private static StartLiveTailResponseHandler
getStartLiveTailResponseStreamHandler(
    AtomicReference<Subscription> subscriptionAtomicReference) {
    return StartLiveTailResponseHandler.builder()
        .onResponse(r -> System.out.println("Received initial response"))
```



```

        .onError(throwable -> {
            CloudWatchLogsException e = (CloudWatchLogsException)
throwable.getCause();
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        })
        .subscriber(() -> new FlowableSubscriber<>() {
            @Override
            public void onSubscribe(@NonNull Subscription s) {
                subscriptionAtomicReference.set(s);
                s.request(Long.MAX_VALUE);
            }

            @Override
            public void onNext(StartLiveTailResponseStream event) {
                if (event instanceof LiveTailSessionStart) {
                    LiveTailSessionStart sessionStart = (LiveTailSessionStart)
event;

                    System.out.println(sessionStart);
                } else if (event instanceof LiveTailSessionUpdate) {
                    LiveTailSessionUpdate sessionUpdate =
(LiveTailSessionUpdate) event;
                    List<LiveTailSessionLogEvent> logEvents =
sessionUpdate.sessionResults();
                    logEvents.forEach(e -> {
                        long timestamp = e.timestamp();
                        Date date = new Date(timestamp);
                        System.out.println "[" + date + "]" + e.message());
                    });
                } else {
                    throw CloudWatchLogsException.builder().message("Unknown
event type").build();
                }
            }

            @Override
            public void onError(Throwable throwable) {
                System.out.println(throwable.getMessage());
                System.exit(1);
            }

            @Override
            public void onComplete() {
                System.out.println("Completed Streaming Session");
            }
        });

```

```

        }
    })
    .build();
}

```

Starten Sie die Live-Tail-Sitzung.

```

CloudWatchLogsAsyncClient cloudWatchLogsAsyncClient =
    CloudWatchLogsAsyncClient.builder()
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

StartLiveTailRequest request =
    StartLiveTailRequest.builder()
        .logGroupIdentifiers(logGroupIdentifiers)
        .logStreamNames(logStreamNames)
        .logEventFilterPattern(logEventFilterPattern)
        .build();

/* Create a reference to store the subscription */
final AtomicReference<Subscription> subscriptionAtomicReference = new
AtomicReference<>(null);

cloudWatchLogsAsyncClient.startLiveTail(request,
getStartLiveTailResponseStreamHandler(subscriptionAtomicReference));

```

Halten Sie die Live-Tail-Sitzung an, nachdem ein Zeitraum verstrichen ist.

```

/* Set a timeout for the session and cancel the subscription. This will:
 * 1). Close the stream
 * 2). Stop the Live Tail session
 */
try {
    Thread.sleep(10000);
} catch (InterruptedException e) {
    throw new RuntimeException(e);
}
if (subscriptionAtomicReference.get() != null) {
    subscriptionAtomicReference.get().cancel();
    System.out.println("Subscription to stream closed");
}

```

- Weitere API-Informationen finden Sie unter [StartLiveTail](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beispiele für Amazon Cognito Identity unter Verwendung von SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon Cognito Identity verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Themen

- [Aktionen](#)

## Aktionen

### Erstellen eines Identitäten-Pools

Das folgende Codebeispiel zeigt, wie Sie einen Amazon Cognito-Identitätspool erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
    software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolRequest;
import
    software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateIdentityPool {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <identityPoolName>\s

            Where:
                identityPoolName - The name to give your identity pool.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoolName = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String identityPoolId = createIdPool(cognitoClient, identityPoolName);
        System.out.println("Unity pool ID " + identityPoolId);
        cognitoClient.close();
    }

    public static String createIdPool(CognitoIdentityClient cognitoClient, String
identityPoolName) {
        try {
```

```
        CreateIdentityPoolRequest poolRequest =
CreateIdentityPoolRequest.builder()
    .allowUnauthenticatedIdentities(false)
    .identityPoolName(identityPoolName)
    .build();

        CreateIdentityPoolResponse response =
cognitoClient.createIdentityPool(poolRequest);
        return response.identityPoolId();

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [CreateIdentityPool](#)
  - [ListIdentityPools](#)

## Löschen eines Identitätspools

Das folgende Codebeispiel zeigt, wie Sie einen Amazon Cognito-Identitätspool löschen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
software.amazon.awssdk.services.cognitoidentity.model.DeleteIdentityPoolRequest;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteIdentityPool {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <identityPoolId>\s

            Where:
                identityPoolId - The Id value of your identity pool.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoolId = args[0];
        CognitoIdentityClient cognitoIdClient = CognitoIdentityClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        deleteIdPool(cognitoIdClient, identityPoolId);
        cognitoIdClient.close();
    }

    public static void deleteIdPool(CognitoIdentityClient cognitoIdClient, String
identityPoolId) {
        try {

            DeleteIdentityPoolRequest identityPoolRequest =
DeleteIdentityPoolRequest.builder()
                .identityPoolId(identityPoolId)
                .build();
```

```

        cognitoIdClient.deleteIdentityPool(identityPoolRequest);
        System.out.println("Done");

    } catch (AwsServiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Weitere API-Informationen finden Sie unter [DeleteIdentityPool](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Anfordern von Anmeldeinformationen für eine Identität

Das folgende Code-Beispiel veranschaulicht, wie Sie Anmeldeinformationen für eine Amazon-Cognito-Identität erhalten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
    software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityRequest;
import
    software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 */

```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class GetIdentityCredentials {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <identityId>\s

            Where:
                identityId - The Id of an existing identity in the format
REGION:GUID.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityId = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
            .region(Region.US_EAST_1)
            .build();

        getCredsForIdentity(cognitoClient, identityId);
        cognitoClient.close();
    }

    public static void getCredsForIdentity(CognitoIdentityClient cognitoClient,
String identityId) {
        try {
            GetCredentialsForIdentityRequest getCredentialsForIdentityRequest =
GetCredentialsForIdentityRequest
                .builder()
                .identityId(identityId)
                .build();

            GetCredentialsForIdentityResponse response = cognitoClient
                .getCredentialsForIdentity(getCredentialsForIdentityRequest);
            System.out.println(
```



```

        "Identity ID " + response.identityId() + ", Access key ID " +
response.credentials().accessKeyId());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Weitere API-Informationen finden Sie unter [GetCredentialsForIdentity](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Auflisten von Identitätspools

Das folgende Codebeispiel zeigt, wie Sie eine Liste von Amazon Cognito-Identitätspools abrufen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsRequest;
import
software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsResponse;
import
software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */

```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ListIdentityPools {
    public static void main(String[] args) {
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listIdPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listIdPools(CognitoIdentityClient cognitoClient) {
        try {
            ListIdentityPoolsRequest poolsRequest =
ListIdentityPoolsRequest.builder()
                .maxResults(15)
                .build();

            ListIdentityPoolsResponse response =
cognitoClient.listIdentityPools(poolsRequest);
            response.identityPools().forEach(pool -> {
                System.out.println("Pool ID: " + pool.identityPoolId());
                System.out.println("Pool name: " + pool.identityPoolName());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [CreateIdentityPool](#)
  - [ListIdentityPools](#)

## Beispiele für Amazon Cognito-Identitätsanbieter mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x 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 Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Erste Schritte

#### Hello Amazon Cognito

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon Cognito.

#### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
        .region(Region.US_EAST_1)
        .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient cognitoClient)
    {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                .maxResults(10)
                .build();

            ListUserPoolsResponse response = cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID " +
userpool.id());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [ListUserPools](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Themen

- [Aktionen](#)
- [Szenarien](#)

## Aktionen

### Bestätigen eines Benutzers

Das folgende Codebeispiel zeigt, wie Sie einen Amazon Cognito-Benutzer bestätigen.

SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
    String userName) {
    try {
        ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
            .clientId(clientId)
            .confirmationCode(code)
            .username(userName)
            .build();

        identityProviderClient.confirmSignUp(signUpRequest);
        System.out.println(userName + " was confirmed");

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [ConfirmSignUp](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen eines Benutzerpools

Das folgende Codebeispiel zeigt, wie Sie einen Amazon Cognito-Benutzerpool erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUserPool {
    public static void main(String[] args) {

        final String usage = ""

                Usage:
                    <userPoolName>\s

                Where:
                    userPoolName - The name to give your user pool when it's
created.

        "";
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolName = args[0];
        CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String id = createPool(cognitoClient, userPoolName);
        System.out.println("User pool ID: " + id);
        cognitoClient.close();
    }

    public static String createPool(CognitoIdentityProviderClient cognitoClient,
String userPoolName) {
        try {
            CreateUserPoolRequest request = CreateUserPoolRequest.builder()
                .poolName(userPoolName)
                .build();

            CreateUserPoolResponse response = cognitoClient.createUserPool(request);
            return response.userPool().id();

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Weitere API-Informationen finden Sie unter [CreateUserPool](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen eines App-Clients

Das folgende Code-Beispiel veranschaulicht, wie Sie eine Amazon-Cognito-Benutzerpool-Client-App erstellen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientResponse;

/**
 * A user pool client app is an application that authenticates with Amazon
 * Cognito user pools.
 * When you create a user pool, you can configure app clients that allow mobile
 * or web applications
 * to call API operations to authenticate users, manage user attributes and
 * profiles,
 * and implement sign-up and sign-in flows.
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUserPoolClient {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <clientName> <userPoolId>\s

                Where:
                clientName - The name for the user pool client to create.
```



```
        userPoolId - The ID for the user pool.
        """);

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String clientName = args[0];
    String userPoolId = args[1];
    CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
        .region(Region.US_EAST_1)
        .build();

    createPoolClient(cognitoClient, clientName, userPoolId);
    cognitoClient.close();
}

public static void createPoolClient(CognitoIdentityProviderClient cognitoClient,
String clientName,
    String userPoolId) {
    try {
        CreateUserPoolClientRequest request =
CreateUserPoolClientRequest.builder()
            .clientName(clientName)
            .userPoolId(userPoolId)
            .build();

        CreateUserPoolClientResponse response =
cognitoClient.createUserPoolClient(request);
        System.out.println("User pool " + response.userPoolClient().clientName()
+ " created. ID: "
            + response.userPoolClient().clientId());


    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [CreateUserPoolClient](#) in der APIAWS SDK for Java 2.x -Referenz für .

Abruf eines Tokens, um eine MFA-Anwendung einem Benutzer zuzuordnen

Das folgende Codebeispiel zeigt, wie Sie ein Token abrufen, um eine MFA-Anwendung einem Amazon Cognito-Benutzer zuzuordnen.

SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
    AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
        .session(session)
        .build();

    AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
        .associateSoftwareToken(softwareTokenRequest);
    String secretCode = tokenResponse.secretCode();
    System.out.println("Enter this token into Google Authenticator");
    System.out.println(secretCode);
    return tokenResponse.session();
}
```

- Weitere API-Informationen finden Sie unter [AssociateSoftwareToken](#) in der APIAWS SDK for Java 2.x -Referenz für .

Abrufen von Informationen über einen Benutzer

Das folgende Codebeispiel zeigt, wie Sie Informationen zu einem Amazon Cognito-Benutzer abrufen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName,
    String poolId) {
    try {
        AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
            .username(userName)
            .userPoolId(poolId)
            .build();

        AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
        System.out.println("User status " + response.userStatusAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [AdminGetUser](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten der Benutzerpools

Das folgende Code-Beispiel veranschaulicht, wie Amazon Cognito-Benutzerpools aufgelistet werden.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
        CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient cognitoClient)
    {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                .maxResults(10)
```

```

        .build();

        ListUserPoolsResponse response = cognitoClient.listUserPools(request);
        response.userPools().forEach(userpool -> {
            System.out.println("User pool " + userpool.name() + ", User ID " +
                userpool.id());
        });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- Weitere API-Informationen finden Sie unter [ListUserPools](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Benutzern

Das folgende Codebeispiel zeigt, wie Sie Amazon Cognito-Benutzer auflisten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersResponse;

/**

```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ListUsers {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <userPoolId>\s

            Where:
                userPoolId - The ID given to your user pool when it's created.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolId = args[0];
        CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllUsers(cognitoClient, userPoolId);
        listUsersFilter(cognitoClient, userPoolId);
        cognitoClient.close();
    }

    public static void listAllUsers(CognitoIdentityProviderClient cognitoClient,
String userPoolId) {
        try {
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                .userPoolId(userPoolId)
                .build();

            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
```

```
        System.out.println("User " + user.username() + " Status " +
user.userStatus() + " Created "
        + user.userCreateDate());
    });

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Shows how to list users by using a filter.
public static void listUsersFilter(CognitoIdentityProviderClient cognitoClient,
String userPoolId) {

    try {
        String filter = "email = \"tblue@noserver.com\"";
        ListUsersRequest usersRequest = ListUsersRequest.builder()
            .userPoolId(userPoolId)
            .filter(filter)
            .build();

        ListUsersResponse response = cognitoClient.listUsers(usersRequest);
        response.users().forEach(user -> {
            System.out.println("User with filter applied " + user.username() + "
Status " + user.userStatus()
            + " Created " + user.userCreateDate());
        });

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [ListUsers](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erneutes Senden eines Bestätigungscode

Das folgende Codebeispiel zeigt, wie Sie einen Amazon Cognito-Bestätigungscode erneut senden.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
    String userName) {
    try {
        ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
            .clientId(clientId)
            .username(userName)
            .build();

        ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
        System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [ResendConfirmationCode](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Reagieren auf eine Amazon Cognito SRP-Authentifizierungsherausforderung

Das folgende Codebeispiel zeigt, wie Sie auf eine Amazon Cognito-Authentifizierungsaufforderung reagieren.



## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
// Respond to an authentication challenge.
public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
    String userName, String clientId, String mfaCode, String session) {
    System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
    Map<String, String> challengeResponses = new HashMap<>();

    challengeResponses.put("USERNAME", userName);
    challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

    AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
        .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
        .clientId(clientId)
        .challengeResponses(challengeResponses)
        .session(session)
        .build();

    AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
        .adminRespondToAuthChallenge(respondToAuthChallengeRequest);
    System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
        + respondToAuthChallengeResult.authenticationResult());
}
```

- Weitere API-Informationen finden Sie unter [AdminRespondToAuthChallenge](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Registrieren eines Benutzers

Das folgende Codebeispiel zeigt, wie Sie einen Benutzer mit Amazon Cognito registrieren.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName,
    String password, String email) {
    AttributeType userAttrs = AttributeType.builder()
        .name("email")
        .value(email)
        .build();

    List<AttributeType> userAttrsList = new ArrayList<>();
    userAttrsList.add(userAttrs);
    try {
        SignUpRequest signUpRequest = SignUpRequest.builder()
            .userAttributes(userAttrsList)
            .username(userName)
            .clientId(clientId)
            .password(password)
            .build();

        identityProviderClient.signUp(signUpRequest);
        System.out.println("User has been signed up ");

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [SignUp](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Starten der Authentifizierung mit Administratoranmeldeinformationen

Das folgende Codebeispiel zeigt, wie Sie die Authentifizierung mit Amazon Cognito und Administratoranmeldeinformationen starten.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
             String clientId, String userName, String password, String userPoolId) {
    try {
        Map<String, String> authParameters = new HashMap<>();
        authParameters.put("USERNAME", userName);
        authParameters.put("PASSWORD", password);

        AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                        .clientId(clientId)
                        .userPoolId(userPoolId)
                        .authParameters(authParameters)
                        .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                        .build();

        AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
        System.out.println("Result Challenge is : " + response.challengeName());
        return response;

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Weitere API-Informationen finden Sie unter [AdminInitiateAuth](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Verifizieren einer MFA-Anwendung mit einem Benutzer

Das folgende Codebeispiel zeigt, wie Sie eine MFA-Anwendung mit einem Amazon Cognito-Benutzer verifizieren.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
// Verify the TOTP and register for MFA.
public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
    try {
        VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
            .userCode(code)
            .session(session)
            .build();

        VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
        System.out.println("The status of the token is " +
verifyResponse.statusAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [VerifySoftwareToken](#) in der APIAWS SDK for Java 2.x -Referenz für .

## 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.
- Einrichten der Multi-Faktor-Authentifizierung durch Zuordnung einer MFA-Anwendung zu dem Benutzer.
- Anmelden unter Verwendung eines Passworts und eines MFA-Codes.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChallengeRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChallengeResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenResponse;
import software.amazon.awssdk.services.cognitoidentityprovider.model.AttributeType;
```

```
import software.amazon.awssdk.services.cognitoidentityprovider.model.AuthFlowType;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ChallengeNameType;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ConfirmSignUpRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRequest
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeResponse
import software.amazon.awssdk.services.cognitoidentityprovider.model.SignUpRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenResponse;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * TIP: To set up the required user pool, run the AWS Cloud Development Kit (AWS
 * CDK) script provided in this GitHub repo at
 * resources/cdk/cognito\_scenario\_user\_pool\_with\_mfa.
 *
 * This code example performs the following operations:
 *
 * 1. Invokes the signUp method to sign up a user.
 * 2. Invokes the adminGetUser method to get the user's confirmation status.
 * 3. Invokes the ResendConfirmationCode method if the user requested another
 * code.
 * 4. Invokes the confirmSignUp method.
 * 5. Invokes the AdminInitiateAuth to sign in. This results in being prompted
```

```

* to set up TOTP (time-based one-time password). (The response is
* "ChallengeName": "MFA_SETUP").
* 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
* key. This can be used with Google Authenticator.
* 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
* MFA.
* 8. Invokes the AdminInitiateAuth to sign in again. This results in being
* prompted to submit a TOTP (Response: "ChallengeName": "SOFTWARE_TOKEN_MFA").
* 9. Invokes the AdminRespondToAuthChallenge to get back a token.
*/

```

```

public class CognitoMVP {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws NoSuchAlgorithmException,
InvalidKeyException {
        final String usage = ""

            Usage:
                <clientId> <poolId>

            Where:
                clientId - The app client Id value that you can get from the AWS
CDK script.
                poolId - The pool Id that you can get from the AWS CDK script.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientId = args[0];
        String poolId = args[1];
        CognitoIdentityProviderClient identityProviderClient =
CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Cognito example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);

```

```
System.out.println("*** Enter your user name");
Scanner in = new Scanner(System.in);
String userName = in.nextLine();

System.out.println("*** Enter your password");
String password = in.nextLine();

System.out.println("*** Enter your email");
String email = in.nextLine();

System.out.println("1. Signing up " + userName);
signUp(identityProviderClient, clientId, userName, password, email);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Getting " + userName + " in the user pool");
getAdminUser(identityProviderClient, userName, poolId);

System.out
    .println("*** Conformation code sent to " + userName + ". Would you
like to send a new code? (Yes/No)");
System.out.println(DASHES);

System.out.println(DASHES);
String ans = in.nextLine();

if (ans.compareTo("Yes") == 0) {
    resendConfirmationCode(identityProviderClient, clientId, userName);
    System.out.println("3. Sending a new confirmation code");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Enter confirmation code that was emailed");
String code = in.nextLine();
confirmSignUp(identityProviderClient, clientId, code, userName);
System.out.println("Rechecking the status of " + userName + " in the user
pool");
getAdminUser(identityProviderClient, userName, poolId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Invokes the initiateAuth to sign in");
```



```
        AdminInitiateAuthResponse authResponse =
initiateAuth(identityProviderClient, clientId, userName, password,
            poolId);
        String mySession = authResponse.session();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Invokes the AssociateSoftwareToken method to generate
a TOTP key");
        String newSession = getSecretForAppMFA(identityProviderClient, mySession);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter the 6-digit code displayed in Google
Authenticator");
        String myCode = in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Verify the TOTP and register for MFA");
        verifyTOTP(identityProviderClient, newSession, myCode);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Re-enter a 6-digit code displayed in Google
Authenticator");
        String mfaCode = in.nextLine();
        AdminInitiateAuthResponse authResponse1 =
initiateAuth(identityProviderClient, clientId, userName, password,
            poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Invokes the AdminRespondToAuthChallenge");
        String session2 = authResponse1.session();
        adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
mfaCode, session2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon Cognito operations were successfully
performed");
        System.out.println(DASHES);
    }
```

```
// Respond to an authentication challenge.
public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
    String userName, String clientId, String mfaCode, String session) {
    System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
    Map<String, String> challengeResponses = new HashMap<>();

    challengeResponses.put("USERNAME", userName);
    challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

    AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
    .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
    .clientId(clientId)
    .challengeResponses(challengeResponses)
    .session(session)
    .build();

    AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
    .adminRespondToAuthChallenge(respondToAuthChallengeRequest);
    System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
    + respondToAuthChallengeResult.authenticationResult());
}

// Verify the TOTP and register for MFA.
public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
    try {
        VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
        .userCode(code)
        .session(session)
        .build();

        VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
        System.out.println("The status of the token is " +
verifyResponse.statusAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
  }  
  
  public static AdminInitiateAuthResponse  
  initiateAuth(CognitoIdentityProviderClient identityProviderClient,  
              String clientId, String userName, String password, String userPoolId) {  
    try {  
      Map<String, String> authParameters = new HashMap<>();  
      authParameters.put("USERNAME", userName);  
      authParameters.put("PASSWORD", password);  
  
      AdminInitiateAuthRequest authRequest =  
      AdminInitiateAuthRequest.builder()  
        .clientId(clientId)  
        .userPoolId(userPoolId)  
        .authParameters(authParameters)  
        .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)  
        .build();  
  
      AdminInitiateAuthResponse response =  
      identityProviderClient.adminInitiateAuth(authRequest);  
      System.out.println("Result Challenge is : " + response.challengeName());  
      return response;  
  
    } catch (CognitoIdentityProviderException e) {  
      System.err.println(e.awsErrorDetails().errorMessage());  
      System.exit(1);  
    }  
  
    return null;  
  }  
  
  public static String getSecretForAppMFA(CognitoIdentityProviderClient  
  identityProviderClient, String session) {  
    AssociateSoftwareTokenRequest softwareTokenRequest =  
    AssociateSoftwareTokenRequest.builder()  
      .session(session)  
      .build();  
  
    AssociateSoftwareTokenResponse tokenResponse = identityProviderClient  
      .associateSoftwareToken(softwareTokenRequest);  
    String secretCode = tokenResponse.secretCode();  
    System.out.println("Enter this token into Google Authenticator");  
    System.out.println(secretCode);  
  }  
}
```

```
        return tokenResponse.session();
    }

    public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
        String userName) {
        try {
            ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                .clientId(clientId)
                .confirmationCode(code)
                .username(userName)
                .build();

            identityProviderClient.confirmSignUp(signUpRequest);
            System.out.println(userName + " was confirmed");

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
        String userName) {
        try {
            ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                .clientId(clientId)
                .username(userName)
                .build();

            ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
            System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName,
    String password, String email) {
    AttributeType userAttrs = AttributeType.builder()
        .name("email")
        .value(email)
        .build();

    List<AttributeType> userAttrsList = new ArrayList<>();
    userAttrsList.add(userAttrs);
    try {
        SignUpRequest signUpRequest = SignUpRequest.builder()
            .userAttributes(userAttrsList)
            .username(userName)
            .clientId(clientId)
            .password(password)
            .build();

        identityProviderClient.signUp(signUpRequest);
        System.out.println("User has been signed up ");

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName,
    String poolId) {
    try {
        AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
            .username(userName)
            .userPoolId(poolId)
            .build();

        AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
        System.out.println("User status " + response.userStatusAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}  
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [AdminGetUser](#)
  - [AdminInitiateAuth](#)
  - [AdminRespondToAuthChallenge](#)
  - [AssociateSoftwareToken](#)
  - [ConfirmDevice](#)
  - [ConfirmSignUp](#)
  - [InitiateAuth](#)
  - [ListUsers](#)
  - [ResendConfirmationCode](#)
  - [RespondToAuthChallenge](#)
  - [SignUp](#)
  - [VerifySoftwareToken](#)

## Amazon Comprehend-Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon Comprehend verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Themen

- [Aktionen](#)  
Amazon Comprehend

## Aktionen

### Erstellen eines Dokumentklassifizierers

Das folgende Codebeispiel zeigt, wie Sie einen Amazon Comprehend-Dokumentklassifizierer erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
    software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierRequest;
import
    software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierResponse;
import
    software.amazon.awssdk.services.comprehend.model.DocumentClassifierInputDataConfig;

/**
 * Before running this code example, you can setup the necessary resources, such
 * as the CSV file and IAM Roles, by following this document:
 * https://aws.amazon.com/blogs/machine-learning/building-a-custom-classifier-using-
amazon-comprehend/
 *
 * Also, set up your development environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DocumentClassifierDemo {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <dataAccessRoleArn> <s3Uri> <documentClassifierName>
```

```
        Where:
            dataAccessRoleArn - The ARN value of the role used for this
operation.
            s3Uri - The Amazon S3 bucket that contains the CSV file.
            documentClassifierName - The name of the document classifier.
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String dataAccessRoleArn = args[0];
    String s3Uri = args[1];
    String documentClassifierName = args[2];

    Region region = Region.US_EAST_1;
    ComprehendClient comClient = ComprehendClient.builder()
        .region(region)
        .build();

    createDocumentClassifier(comClient, dataAccessRoleArn, s3Uri,
documentClassifierName);
    comClient.close();
}

public static void createDocumentClassifier(ComprehendClient comClient, String
dataAccessRoleArn, String s3Uri,
    String documentClassifierName) {
    try {
        DocumentClassifierInputDataConfig config =
DocumentClassifierInputDataConfig.builder()
            .s3Uri(s3Uri)
            .build();

        CreateDocumentClassifierRequest createDocumentClassifierRequest =
CreateDocumentClassifierRequest.builder()
            .documentClassifierName(documentClassifierName)
            .dataAccessRoleArn(dataAccessRoleArn)
            .languageCode("en")
            .inputDataConfig(config)
            .build();
    }
}
```



```

        CreateDocumentClassifierResponse createDocumentClassifierResult =
comClient
            .createDocumentClassifier(createDocumentClassifierRequest);
        String documentClassifierArn =
createDocumentClassifierResult.documentClassifierArn();
        System.out.println("Document Classifier ARN: " + documentClassifierArn);

    } catch (ComprehendException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

- Weitere API-Informationen finden Sie unter [CreateDocumentClassifier](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erkennen von Entitäten in einem Dokument

Das folgende Codebeispiel zeigt, wie Entitäten in einem Dokument mit Amazon Comprehend erkannt werden.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesResponse;
import software.amazon.awssdk.services.comprehend.model.Entity;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development

```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DetectEntities {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
        July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
        blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
        Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
            .region(region)
            .build();

        System.out.println("Calling DetectEntities");
        detectAllEntities(comClient, text);
        comClient.close();
    }

    public static void detectAllEntities(ComprehendClient comClient, String text) {
        try {
            DetectEntitiesRequest detectEntitiesRequest =
            DetectEntitiesRequest.builder()
                .text(text)
                .languageCode("en")
                .build();

            DetectEntitiesResponse detectEntitiesResult =
            comClient.detectEntities(detectEntitiesRequest);
            List<Entity> entList = detectEntitiesResult.entities();
            for (Entity entity : entList) {
                System.out.println("Entity text is " + entity.text());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [DetectEntities](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erkennen von Schlüsselphrasen in einem Dokument

Das folgende Codebeispiel zeigt, wie Sie Schlüsselphrasen in einem Dokument mit Amazon Comprehend erkennen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesResponse;
import software.amazon.awssdk.services.comprehend.model.KeyPhrase;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectKeyPhrases {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
        July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
        blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
        Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
```

```
ComprehendClient comClient = ComprehendClient.builder()
    .region(region)
    .build();

System.out.println("Calling DetectKeyPhrases");
detectAllKeyPhrases(comClient, text);
comClient.close();
}

public static void detectAllKeyPhrases(ComprehendClient comClient, String text)
{
    try {
        DetectKeyPhrasesRequest detectKeyPhrasesRequest =
DetectKeyPhrasesRequest.builder()
            .text(text)
            .languageCode("en")
            .build();

        DetectKeyPhrasesResponse detectKeyPhrasesResult =
comClient.detectKeyPhrases(detectKeyPhrasesRequest);
        List<KeyPhrase> phraseList = detectKeyPhrasesResult.keyPhrases();
        for (KeyPhrase keyPhrase : phraseList) {
            System.out.println("Key phrase text is " + keyPhrase.text());
        }

    } catch (ComprehendException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [DetectKeyPhrases](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erkennen syntaktischer Elemente eines Dokuments

Das folgende Codebeispiel zeigt, wie Sie syntaktische Elemente eines Dokuments mit Amazon Comprehend erkennen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxResponse;
import software.amazon.awssdk.services.comprehend.model.SyntaxToken;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectSyntax {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
        July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
        blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
        Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
            .region(region)
            .build();

        System.out.println("Calling DetectSyntax");
        detectAllSyntax(comClient, text);
        comClient.close();
    }

    public static void detectAllSyntax(ComprehendClient comClient, String text) {
        try {
```

```
        DetectSyntaxRequest detectSyntaxRequest = DetectSyntaxRequest.builder()
            .text(text)
            .languageCode("en")
            .build();

        DetectSyntaxResponse detectSyntaxResult =
comClient.detectSyntax(detectSyntaxRequest);
        List<SyntaxToken> syntaxTokens = detectSyntaxResult.syntaxTokens();
        for (SyntaxToken token : syntaxTokens) {
            System.out.println("Language is " + token.text());
            System.out.println("Part of speech is " +
token.partOfSpeech().tagAsString());
        }

    } catch (ComprehendException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [DetectSyntax](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erkennen der dominanten Sprache in einem Dokument

Das folgende Codebeispiel zeigt, wie Sie die vorherrschende Sprache in einem Dokument mit Amazon Comprehend erkennen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
```

```
import
    software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageRequest;
import
    software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageResponse;
import software.amazon.awssdk.services.comprehend.model.DominantLanguage;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectLanguage {
    public static void main(String[] args) {
        // Specify French text - "It is raining today in Seattle".
        String text = "Il pleut aujourd'hui à Seattle";
        Region region = Region.US_EAST_1;

        ComprehendClient comClient = ComprehendClient.builder()
            .region(region)
            .build();

        System.out.println("Calling DetectDominantLanguage");
        detectTheDominantLanguage(comClient, text);
        comClient.close();
    }

    public static void detectTheDominantLanguage(ComprehendClient comClient, String
text) {
        try {
            DetectDominantLanguageRequest request =
DetectDominantLanguageRequest.builder()
                .text(text)
                .build();

            DetectDominantLanguageResponse resp =
comClient.detectDominantLanguage(request);
            List<DominantLanguage> allLanList = resp.languages();
            for (DominantLanguage lang : allLanList) {
                System.out.println("Language is " + lang.languageCode());
            }
        }
    }
}
```

```
        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [DetectDominantLanguage](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erkennen der Stimmung eines Dokuments

Das folgende Codebeispiel zeigt, wie Sie die Stimmung eines Dokuments mit Amazon Comprehend erkennen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectSentiment {
    public static void main(String[] args) {
```



```

    String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
    July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
    blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
    Seattle - based companies are Starbucks and Boeing.";
    Region region = Region.US_EAST_1;
    ComprehendClient comClient = ComprehendClient.builder()
        .region(region)
        .build();

    System.out.println("Calling DetectSentiment");
    detectSentiments(comClient, text);
    comClient.close();
}

public static void detectSentiments(ComprehendClient comClient, String text) {
    try {
        DetectSentimentRequest detectSentimentRequest =
        DetectSentimentRequest.builder()
            .text(text)
            .languageCode("en")
            .build();

        DetectSentimentResponse detectSentimentResult =
        comClient.detectSentiment(detectSentimentRequest);
        System.out.println("The Neutral value is " +
        detectSentimentResult.sentimentScore().neutral());

    } catch (ComprehendException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

- Weitere API-Informationen finden Sie unter [DetectSentiment](#) in der APIAWS SDK for Java 2.x - Referenz für .

## DynamoDB-Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit DynamoDB verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

## Erste Schritte

### Hallo DynamoDB

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit DynamoDB.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTables {
    public static void main(String[] args) {
        System.out.println("Listing your Amazon DynamoDB tables:\n");
    }
}
```

```
Region region = Region.US_EAST_1;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();
listAllTables(ddb);
ddb.close();
}

public static void listAllTables(DynamoDbClient ddb) {
    boolean moreTables = true;
    String lastName = null;

    while (moreTables) {
        try {
            ListTablesResponse response = null;
            if (lastName == null) {
                ListTablesRequest request = ListTablesRequest.builder().build();
                response = ddb.listTables(request);
            } else {
                ListTablesRequest request = ListTablesRequest.builder()
                    .exclusiveStartTableName(lastName).build();
                response = ddb.listTables(request);
            }

            List<String> tableNames = response.tableNames();
            if (tableNames.size() > 0) {
                for (String curName : tableNames) {
                    System.out.format("* %s\n", curName);
                }
            } else {
                System.out.println("No tables found!");
                System.exit(0);
            }

            lastName = response.lastEvaluatedTableName();
            if (lastName == null) {
                moreTables = false;
            }
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

```
        System.out.println("\nDone!");
    }
}
```

- Weitere API-Informationen finden Sie unter [ListTables](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Themen

- [Aktionen](#)
- [Szenarien](#)

## Aktionen

### Erstellen einer Tabelle

Im folgenden Codebeispiel wird gezeigt, wie eine DynamoDB-Tabelle erstellt wird.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTable {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName> <key>

            Where:
                tableName - The Amazon DynamoDB table to create (for example,
Music3).
                key - The key for the Amazon DynamoDB table (for example,
Artist).

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        System.out.println("Creating an Amazon DynamoDB table " + tableName + " with
a simple primary key: " + key);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        String result = createTable(ddb, tableName, key);
        System.out.println("New table is " + result);
        ddb.close();
    }

    public static String createTable(DynamoDbClient ddb, String tableName, String
key) {
```

```
DynamoDbWaiter dbWaiter = ddb.waiter();
CreateTableRequest request = CreateTableRequest.builder()
    .attributeDefinitions(AttributeDefinition.builder()
        .attributeName(key)
        .attributeType(ScalarAttributeType.S)
        .build())
    .keySchema(KeySchemaElement.builder()
        .attributeName(key)
        .keyType(KeyType.HASH)
        .build())
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(10L)
        .writeCapacityUnits(10L)
        .build())
    .tableName(tableName)
    .build();

String newTable;
try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    newTable = response.tableDescription().tableName();
    return newTable;

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}
}
```

- Weitere API-Informationen finden Sie unter [CreateTable](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen einer Tabelle

Im folgenden Codebeispiel wird gezeigt, wie eine DynamoDB-Tabelle gelöscht wird.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteTable {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName>

            Where:
                tableName - The Amazon DynamoDB table to delete (for example,
Music3).

            **Warning** This program will delete the table that you specify!
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String tableName = args[0];
    System.out.format("Deleting the Amazon DynamoDB table %s...\n", tableName);
    Region region = Region.US_EAST_1;
    DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build();

    deleteDynamoDBTable(ddb, tableName);
    ddb.close();
}

public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}
}
```

- Weitere API-Informationen finden Sie unter [DeleteTable](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen eines Elements aus einer Tabelle

Im folgenden Codebeispiel wird gezeigt, wie ein Element aus einer DynamoDB-Tabelle gelöscht wird.



## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteItem {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <tableName> <key> <keyval>

                Where:
                tableName - The Amazon DynamoDB table to delete the item from
(for example, Music3).
                key - The key used in the Amazon DynamoDB table (for example,
Artist).\s
                keyval - The key value that represents the item to delete (for
example, Famous Band).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```

    }

    String tableName = args[0];
    String key = args[1];
    String keyVal = args[2];
    System.out.format("Deleting item \"%s\" from %s\n", keyVal, tableName);
    Region region = Region.US_EAST_1;
    DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build();

    deleteDynamoDBItem(ddb, tableName, key, keyVal);
    ddb.close();
}

public static void deleteDynamoDBItem(DynamoDbClient ddb, String tableName,
String key, String keyVal) {
    HashMap<String, AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal)
        .build());

    DeleteItemRequest deleteReq = DeleteItemRequest.builder()
        .tableName(tableName)
        .key(keyToGet)
        .build();

    try {
        ddb.deleteItem(deleteReq);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

- Weitere API-Informationen finden Sie unter [DeleteItem](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Abrufen eines Element-Batches

Im folgenden Codebeispiel wird gezeigt, wie ein DynamoDB-Element-Batch abgerufen wird.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

zeigt, wie Batch-Elemente mit dem Service-Client abgerufen werden.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.KeysAndAttributes;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class BatchReadItems {
    public static void main(String[] args){
        final String usage = ""

                Usage:
                <tableName>

                Where:
                tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
                .region(region)
```

```
        .build();

    getBatchItems(dynamoDbClient, tableName);
}

public static void getBatchItems(DynamoDbClient dynamoDbClient, String
tableName) {
    // Define the primary key values for the items you want to retrieve.
    Map<String, AttributeValue> key1 = new HashMap<>();
    key1.put("Artist", AttributeValue.builder().s("Artist1").build());

    Map<String, AttributeValue> key2 = new HashMap<>();
    key2.put("Artist", AttributeValue.builder().s("Artist2").build());

    // Construct the batchGetItem request.
    Map<String, KeysAndAttributes> requestItems = new HashMap<>();
    requestItems.put(tableName, KeysAndAttributes.builder()
        .keys(List.of(key1, key2))
        .projectionExpression("Artist, SongTitle")
        .build());

    BatchGetItemRequest batchGetItemRequest = BatchGetItemRequest.builder()
        .requestItems(requestItems)
        .build();

    // Make the batchGetItem request.
    BatchGetItemResponse batchGetItemResponse =
dynamoDbClient.batchGetItem(batchGetItemRequest);

    // Extract and print the retrieved items.
    Map<String, List<Map<String, AttributeValue>>> responses =
batchGetItemResponse.responses();
    if (responses.containsKey(tableName)) {
        List<Map<String, AttributeValue>> musicItems = responses.get(tableName);
        for (Map<String, AttributeValue> item : musicItems) {
            System.out.println("Artist: " + item.get("Artist").s() +
                ", SongTitle: " + item.get("SongTitle").s());
        }
    } else {
        System.out.println("No items retrieved.");
    }
}
}
```

zeigt, wie Batch-Elemente mit dem Service-Client und einem Paginator abgerufen werden.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.KeysAndAttributes;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class BatchGetItemsPaginator {

    public static void main(String[] args){
        final String usage = ""

            Usage:
                <tableName>

            Where:
                tableName - The Amazon DynamoDB table (for example, Music).\s
            """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();

        getBatchItemsPaginator(dynamoDbClient, tableName) ;
    }

    public static void getBatchItemsPaginator(DynamoDbClient dynamoDbClient, String
tableName) {
        // Define the primary key values for the items you want to retrieve.
        Map<String, AttributeValue> key1 = new HashMap<>();
        key1.put("Artist", AttributeValue.builder().s("Artist1").build());

        Map<String, AttributeValue> key2 = new HashMap<>();
        key2.put("Artist", AttributeValue.builder().s("Artist2").build());
```

```

// Construct the batchGetItem request.
Map<String, KeysAndAttributes> requestItems = new HashMap<>();
requestItems.put(tableName, KeysAndAttributes.builder()
    .keys(List.of(key1, key2))
    .projectionExpression("Artist, SongTitle")
    .build());

BatchGetItemRequest batchGetItemRequest = BatchGetItemRequest.builder()
    .requestItems(requestItems)
    .build();

// Use batchGetItemPaginator for paginated requests.
dynamoDbClient.batchGetItemPaginator(batchGetItemRequest).stream()
    .flatMap(response -> response.responses().getOrDefault(tableName,
Collections.emptyList()).stream())
    .forEach(item -> {
        System.out.println("Artist: " + item.get("Artist").s() +
            ", SongTitle: " + item.get("SongTitle").s());
    });
}
}

```

- Weitere API-Informationen finden Sie unter [BatchGetItem](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Abrufen eines Elements aus einer Tabelle

Im folgenden Codebeispiel wird gezeigt, wie ein Element aus einer DynamoDB-Tabelle abgerufen wird.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Ruft ein Element aus einer Tabelle mithilfe der ab DynamoDbClient.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To get an item from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client, see the EnhancedGetItem example.
 */
public class GetItem {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName> <key> <keyVal>

            Where:
                tableName - The Amazon DynamoDB table from which an item is
retrieved (for example, Music3).\s
                key - The key used in the Amazon DynamoDB table (for example,
Artist).\s
                keyval - The key value that represents the item to get (for
example, Famous Band).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
```

```
String key = args[1];
String keyVal = args[2];
System.out.format("Retrieving item \"%s\" from \"%s\"\n", keyVal,
tableName);
Region region = Region.US_EAST_1;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();

getDynamoDBItem(ddb, tableName, key, keyVal);
ddb.close();
}

public static void getDynamoDBItem(DynamoDbClient ddb, String tableName, String
key, String keyVal) {
    HashMap<String, AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal)
        .build());

    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName(tableName)
        .build();

    try {
        // If there is no matching item, GetItem does not return any data.
        Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();
        if (returnedItem.isEmpty())
            System.out.format("No item found with the key %s!\n", key);
        else {
            Set<String> keys = returnedItem.keySet();
            System.out.println("Amazon DynamoDB table attributes: \n");
            for (String key1 : keys) {
                System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
            }
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```



```
}
```

- Weitere API-Informationen finden Sie unter [GetItem](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Abrufen von Informationen zu einer Tabelle

Im folgenden Codebeispiel wird gezeigt, wie Informationen zu einer DynamoDB-Tabelle abgerufen werden.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import
    software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughputDescription;
import software.amazon.awssdk.services.dynamodb.model.TableDescription;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeTable {
    public static void main(String[] args) {
        final String usage = ""
```

```

        Usage:
            <tableName>

        Where:
            tableName - The Amazon DynamoDB table to get information about
(for example, Music3).
            """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String tableName = args[0];
    System.out.format("Getting description for %s\n\n", tableName);
    Region region = Region.US_EAST_1;
    DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build();

    describeDynamoDBTable(ddb, tableName);
    ddb.close();
}

public static void describeDynamoDBTable(DynamoDbClient ddb, String tableName) {
    DescribeTableRequest request = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        TableDescription tableInfo = ddb.describeTable(request).table();
        if (tableInfo != null) {
            System.out.format("Table name   : %s\n", tableInfo.tableName());
            System.out.format("Table ARN   : %s\n", tableInfo.tableArn());
            System.out.format("Status      : %s\n", tableInfo.tableStatus());
            System.out.format("Item count  : %d\n", tableInfo.itemCount());
            System.out.format("Size (bytes): %d\n", tableInfo.tableSizeBytes());

            ProvisionedThroughputDescription throughputInfo =
tableInfo.provisionedThroughput();
            System.out.println("Throughput");
            System.out.format("  Read Capacity : %d\n",
throughputInfo.readCapacityUnits());

```

```
        System.out.format("  Write Capacity: %d\n",
throughputInfo.writeCapacityUnits());

        List<AttributeDefinition> attributes =
tableInfo.attributeDefinitions();
        System.out.println("Attributes");
        for (AttributeDefinition a : attributes) {
            System.out.format("  %s (%s)\n", a.attributeName(),
a.attributeType());
        }
    }
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println("\nDone!");
}
```

- Weitere API-Informationen finden Sie unter [DescribeTable](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Tabellen

Im folgenden Codebeispiel wird gezeigt, wie DynamoDB-Tabellen aufgelistet werden.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTables {
    public static void main(String[] args) {
        System.out.println("Listing your Amazon DynamoDB tables:\n");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();
        listAllTables(ddb);
        ddb.close();
    }

    public static void listAllTables(DynamoDbClient ddb) {
        boolean moreTables = true;
        String lastName = null;

        while (moreTables) {
            try {
                ListTablesResponse response = null;
                if (lastName == null) {
                    ListTablesRequest request = ListTablesRequest.builder().build();
                    response = ddb.listTables(request);
                } else {
                    ListTablesRequest request = ListTablesRequest.builder()
                        .exclusiveStartTableName(lastName).build();
                    response = ddb.listTables(request);
                }

                List<String> tableNames = response.tableNames();
                if (tableNames.size() > 0) {
                    for (String curName : tableNames) {
                        System.out.format("* %s\n", curName);
                    }
                } else {
                    System.out.println("No tables found!");
                    System.exit(0);
                }
            }
        }
    }
}
```

```
        }

        lastName = response.lastEvaluatedTableName();
        if (lastName == null) {
            moreTables = false;
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
System.out.println("\nDone!");
}
```

- Weitere API-Informationen finden Sie unter [ListTables](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Einfügen eines Elements in eine Tabelle

Im folgenden Codebeispiel wird gezeigt, wie Sie ein Element in eine DynamoDB-Tabelle einfügen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Fügt ein Element mithilfe von in eine Tabelle ein [DynamoDbClient](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.PutItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import java.util.HashMap;
```

```

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To place items into an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client. See the EnhancedPutItem example.
 */
public class PutItem {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                    <tableName> <key> <keyVal> <albumtitle> <albumtitleval> <awards>
<awardsval> <Songtitle> <songtitleval>

                Where:
                    tableName - The Amazon DynamoDB table in which an item is placed
(for example, Music3).
                    key - The key used in the Amazon DynamoDB table (for example,
Artist).
                    keyval - The key value that represents the item to get (for
example, Famous Band).
                    albumTitle - The Album title (for example, AlbumTitle).
                    AlbumTitleValue - The name of the album (for example, Songs
About Life ).
                    Awards - The awards column (for example, Awards).
                    AwardVal - The value of the awards (for example, 10).
                    SongTitle - The song title (for example, SongTitle).
                    SongTitleVal - The value of the song title (for example, Happy
Day).

                **Warning** This program will place an item that you specify into a
table!

                """;

        if (args.length != 9) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}

```

```
String tableName = args[0];
String key = args[1];
String keyVal = args[2];
String albumTitle = args[3];
String albumTitleValue = args[4];
String awards = args[5];
String awardVal = args[6];
String songTitle = args[7];
String songTitleVal = args[8];

Region region = Region.US_EAST_1;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();

putItemInTable(ddb, tableName, key, keyVal, albumTitle, albumTitleValue,
awards, awardVal, songTitle,
    songTitleVal);
System.out.println("Done!");
ddb.close();
}

public static void putItemInTable(DynamoDbClient ddb,
    String tableName,
    String key,
    String keyVal,
    String albumTitle,
    String albumTitleValue,
    String awards,
    String awardVal,
    String songTitle,
    String songTitleVal) {

    HashMap<String, AttributeValue> itemValues = new HashMap<>();
    itemValues.put(key, AttributeValue.builder().s(keyVal).build());
    itemValues.put(songTitle, AttributeValue.builder().s(songTitleVal).build());
    itemValues.put(albumTitle,
AttributeValue.builder().s(albumTitleValue).build());
    itemValues.put(awards, AttributeValue.builder().s(awardVal).build());

    PutItemRequest request = PutItemRequest.builder()
        .tableName(tableName)
        .item(itemValues)
```

```
        .build();

    try {
        PutItemResponse response = ddb.putItem(request);
        System.out.println(tableName + " was successfully updated. The request
id is "
            + response.responseMetadata().requestId());

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

- Weitere API-Informationen finden Sie unter [PutItem](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Abfragen einer Tabelle

Im folgenden Codebeispiel wird gezeigt, wie DynamoDB-Tabellen abgefragt werden.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Fragt eine Tabelle mithilfe von ab [DynamoDbClient](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
```



```
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To query items from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client. See the EnhancedQueryRecords example.
 */
public class Query {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName> <partitionKeyName> <partitionKeyVal>

            Where:
                tableName - The Amazon DynamoDB table to put the item in (for
example, Music3).
                partitionKeyName - The partition key name of the Amazon DynamoDB
table (for example, Artist).
                partitionKeyVal - The value of the partition key that should
match (for example, Famous Band).
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String partitionKeyName = args[1];
        String partitionKeyVal = args[2];

        // For more information about an alias, see:
```

```
// https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Expressions.ExpressionAttributeNames.html
String partitionAlias = "#a";

System.out.format("Querying %s", tableName);
System.out.println("");
Region region = Region.US_EAST_1;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();

int count = queryTable(ddb, tableName, partitionKeyName, partitionKeyVal,
partitionAlias);
System.out.println("There were " + count + " record(s) returned");
ddb.close();
}

public static int queryTable(DynamoDbClient ddb, String tableName, String
partitionKeyName, String partitionKeyVal,
String partitionAlias) {
// Set up an alias for the partition key name in case it's a reserved word.
HashMap<String, String> attrNameAlias = new HashMap<String, String>();
attrNameAlias.put(partitionAlias, partitionKeyName);

// Set up mapping of the partition name with the value.
HashMap<String, AttributeValue> attrValues = new HashMap<>();
attrValues.put(":" + partitionKeyName, AttributeValue.builder()
    .s(partitionKeyVal)
    .build());

QueryRequest queryReq = QueryRequest.builder()
    .tableName(tableName)
    .keyConditionExpression(partitionAlias + " = :" + partitionKeyName)
    .expressionAttributeNames(attrNameAlias)
    .expressionAttributeValues(attrValues)
    .build();

try {
    QueryResponse response = ddb.query(queryReq);
    return response.count();
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

```
    }  
    return -1;  
  }  
}
```

Fragt eine Tabelle mithilfe von `DynamoDbClient` und eines sekundären Index ab.

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;  
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;  
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;  
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;  
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;  
import java.util.HashMap;  
import java.util.Map;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 *  
 * Create the Movies table by running the Scenario example and loading the Movie  
 * data from the JSON file. Next create a secondary  
 * index for the Movies table that uses only the year column. Name the index  
 * year-index. For more information, see:  
 *  
 * https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.html  
 */  
public class QueryItemsUsingIndex {  
    public static void main(String[] args) {  
        String tableName = "Movies";  
        Region region = Region.US_EAST_1;  
        DynamoDbClient ddb = DynamoDbClient.builder()  
            .region(region)  
            .build();  
  
        queryIndex(ddb, tableName);  
        ddb.close();  
    }  
}
```

```
public static void queryIndex(DynamoDbClient ddb, String tableName) {
    try {
        Map<String, String> expressionAttributesNames = new HashMap<>();
        expressionAttributesNames.put("#year", "year");
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        expressionAttributeValues.put(":yearValue",
AttributeValue.builder().n("2013").build());

        QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .indexName("year-index")
            .keyConditionExpression("#year = :yearValue")
            .expressionAttributeNames(expressionAttributesNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        System.out.println("=== Movie Titles ===");
        QueryResponse response = ddb.query(request);
        response.items()
            .forEach(movie -> System.out.println(movie.get("title").s()));

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [Query](#) in der AWS SDK for Java 2.x -API-Referenz.

## Scannen einer Tabelle

Im folgenden Codebeispiel wird gezeigt, wie eine DynamoDB-Tabelle gescannt wird.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Scannt eine Amazon-DynamoDB-Tabelle mit [DynamoDbClient](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;
import java.util.Map;
import java.util.Set;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To scan items from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client, See the EnhancedScanRecords example.
 */

public class DynamoDBScanItems {
    public static void main(String[] args) {

        final String usage = ""

                Usage:
                <tableName>

                Where:
                tableName - The Amazon DynamoDB table to get information from
                (for example, Music3).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        Region region = Region.US_EAST_1;
```

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

scanItems(ddb, tableName);
ddb.close();
}

public static void scanItems(DynamoDbClient ddb, String tableName) {
    try {
        ScanRequest scanRequest = ScanRequest.builder()
            .tableName(tableName)
            .build();

        ScanResponse response = ddb.scan(scanRequest);
        for (Map<String, AttributeValue> item : response.items()) {
            Set<String> keys = item.keySet();
            for (String key : keys) {
                System.out.println("The key name is " + key + "\n");
                System.out.println("The value is " + item.get(key).s());
            }
        }

    } catch (DynamoDbException e) {
        e.printStackTrace();
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [Scan](#) in der AWS SDK for Java 2.x -API-Referenz.

## Aktualisieren eines Elements in einer Tabelle

Im folgenden Codebeispiel wird gezeigt, wie Sie ein Element in einer DynamoDB-Tabelle aktualisieren.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Aktualisiert ein Element in einer Tabelle mithilfe von [DynamoDbClient](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.AttributeAction;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.AttributeValueUpdate;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To update an Amazon DynamoDB table using the AWS SDK for Java V2, its better
 * practice to use the
 * Enhanced Client, See the EnhancedModifyItem example.
 */
public class UpdateItem {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
            <tableName> <key> <keyVal> <name> <updateVal>

        Where:
            tableName - The Amazon DynamoDB table (for example, Music3).
            key - The name of the key in the table (for example, Artist).
            keyVal - The value of the key (for example, Famous Band).
```

name - The name of the column where the value is updated (for example, Awards).  
updateVal - The value used to update an item (for example, 14).

Example:

```
UpdateItem Music3 Artist Famous Band Awards 14
""";
```

```
if (args.length != 5) {
    System.out.println(usage);
    System.exit(1);
}

String tableName = args[0];
String key = args[1];
String keyVal = args[2];
String name = args[3];
String updateVal = args[4];

Region region = Region.US_EAST_1;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();
updateTableItem(ddb, tableName, key, keyVal, name, updateVal);
ddb.close();
}

public static void updateTableItem(DynamoDbClient ddb,
    String tableName,
    String key,
    String keyVal,
    String name,
    String updateVal) {

    HashMap<String, AttributeValue> itemKey = new HashMap<>();
    itemKey.put(key, AttributeValue.builder()
        .s(keyVal)
        .build());

    HashMap<String, AttributeValueUpdate> updatedValues = new HashMap<>();
    updatedValues.put(name, AttributeValueUpdate.builder()
        .value(AttributeValue.builder().s(updateVal).build())
        .action(AttributeAction.PUT)
        .build());
```



```
UpdateItemRequest request = UpdateItemRequest.builder()
    .tableName(tableName)
    .key(itemKey)
    .attributeUpdates(updatedValues)
    .build();

try {
    ddb.updateItem(request);
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println("The Amazon DynamoDB table was updated!");
}
```

- Weitere API-Informationen finden Sie unter [UpdateItem](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Schreiben eines Element-Batches

Im folgenden Codebeispiel wird gezeigt, wie ein DynamoDB-Element-Batch geschrieben wird.

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Fügt mithilfe des Service-Clients viele Elemente in eine Tabelle ein.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchWriteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.BatchWriteItemResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutRequest;
import software.amazon.awssdk.services.dynamodb.model.WriteRequest;
```

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class BatchWriteItems {
    public static void main(String[] args){
        final String usage = ""

            Usage:
                <tableName>

            Where:
                tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();

        addBatchItems(dynamoDbClient, tableName);
    }

    public static void addBatchItems(DynamoDbClient dynamoDbClient, String
tableName) {
        // Specify the updates you want to perform.
        List<WriteRequest> writeRequests = new ArrayList<>();

        // Set item 1.
        Map<String, AttributeValue> item1Attributes = new HashMap<>();
        item1Attributes.put("Artist",
AttributeValue.builder().s("Artist1").build());
        item1Attributes.put("Rating", AttributeValue.builder().s("5").build());
    }
}
```

```
        item1Attributes.put("Comments", AttributeValue.builder().s("Great
song!").build());
        item1Attributes.put("SongTitle",
AttributeValue.builder().s("SongTitle1").build());

writeRequests.add(WriteRequest.builder().putRequest(PutRequest.builder().item(item1Attributes)

        // Set item 2.
        Map<String, AttributeValue> item2Attributes = new HashMap<>();
        item2Attributes.put("Artist",
AttributeValue.builder().s("Artist2").build());
        item2Attributes.put("Rating", AttributeValue.builder().s("4").build());
        item2Attributes.put("Comments", AttributeValue.builder().s("Nice
melody.").build());
        item2Attributes.put("SongTitle",
AttributeValue.builder().s("SongTitle2").build());

writeRequests.add(WriteRequest.builder().putRequest(PutRequest.builder().item(item2Attributes)

        try {
            // Create the BatchWriteItemRequest.
            BatchWriteItemRequest batchWriteItemRequest =
BatchWriteItemRequest.builder()
                .requestItems(Map.of(tableName, writeRequests))
                .build();

            // Execute the BatchWriteItem operation.
            BatchWriteItemResponse batchWriteItemResponse =
dynamoDbClient.batchWriteItem(batchWriteItemRequest);

            // Process the response.
            System.out.println("Batch write successful: " + batchWriteItemResponse);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Fügt mithilfe des erweiterten Clients viele Elemente in eine Tabelle ein.

```
import com.example.dynamodb.Customer;
import com.example.dynamodb.Music;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.Key;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.enhanced.dynamodb.model.BatchWriteItemEnhancedRequest;
import software.amazon.awssdk.enhanced.dynamodb.model.WriteBatch;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/*
 * Before running this code example, create an Amazon DynamoDB table named Customer
 * with these columns:
 * - id - the id of the record that is the key
 * - custName - the customer name
 * - email - the email value
 * - registrationDate - an instant value when the item was added to the table
 *
 * Also, ensure that you have set up your development environment, including your
 * credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EnhancedBatchWriteItems {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();
        DynamoDbEnhancedClient enhancedClient =
DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();
        putBatchRecords(enhancedClient);
        ddb.close();
    }
}
```

```
    }

    public static void putBatchRecords(DynamoDbEnhancedClient enhancedClient) {
        try {
            DynamoDbTable<Customer> customerMappedTable =
enhancedClient.table("Customer",
                        TableSchema.fromBean(Customer.class));
            DynamoDbTable<Music> musicMappedTable =
enhancedClient.table("Music",
                        TableSchema.fromBean(Music.class));
            LocalDate localDate = LocalDate.parse("2020-04-07");
            LocalDateTime localDateTime = localDate.atStartOfDay();
            Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

            Customer record2 = new Customer();
            record2.setCustName("Fred Pink");
            record2.setId("id110");
            record2.setEmail("fredp@noserver.com");
            record2.setRegistrationDate(instant);

            Customer record3 = new Customer();
            record3.setCustName("Susan Pink");
            record3.setId("id120");
            record3.setEmail("spink@noserver.com");
            record3.setRegistrationDate(instant);

            Customer record4 = new Customer();
            record4.setCustName("Jerry orange");
            record4.setId("id101");
            record4.setEmail("jorange@noserver.com");
            record4.setRegistrationDate(instant);

            BatchWriteItemEnhancedRequest batchWriteItemEnhancedRequest
= BatchWriteItemEnhancedRequest
                                .builder()
                                .writeBatches(
WriteBatch.builder(Customer.class) // add items to the Customer

                                // table

                                .mappedTableResource(customerMappedTable)

                                .addPutItem(builder -> builder.item(record2))
```

```

.addPutItem(builder -> builder.item(record3))

.addPutItem(builder -> builder.item(record4))
                                                    .build(),

WriteBatch.builder(Music.class) // delete an item from the Music

    // table

    .mappedTableResource(musicMappedTable)

    .addDeleteItem(builder -> builder.key(

        Key.builder().partitionValue(

            "Famous Band")

            .build()))
                                                    .build())

                                                    .build();

        // Add three items to the Customer table and delete one item
from the Music
        // table.

enhancedClient.batchWriteItem(batchWriteItemEnhancedRequest);
    System.out.println("done");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}

```

- Weitere API-Informationen finden Sie unter [BatchWriteItem](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Szenarien

### Erste Schritte mit Tabellen, Elementen und Abfragen

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 Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Erstellen Sie eine DynamoDB-Tabelle.

```
// Create a table with a Sort key.
public static void createTable(DynamoDbClient ddb, String tableName) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

    // Define attributes.
    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("year")
        .attributeType("N")
        .build());

    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("title")
        .attributeType("S")
        .build());

    ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
```

```
KeySchemaElement key = KeySchemaElement.builder()
    .attributeName("year")
    .keyType(KeyType.HASH)
    .build();

KeySchemaElement key2 = KeySchemaElement.builder()
    .attributeName("title")
    .keyType(KeyType.RANGE)
    .build();

// Add KeySchemaElement objects to the list.
tableKey.add(key);
tableKey.add(key2);

CreateTableRequest request = CreateTableRequest.builder()
    .keySchema(tableKey)
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(10L)
        .writeCapacityUnits(10L)
        .build())
    .attributeDefinitions(attributeDefinitions)
    .tableName(tableName)
    .build();

try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    String newTable = response.tableDescription().tableName();
    System.out.println("The " + newTable + " was successfully created.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```



Erstellen Sie eine Helper-Funktion zum Herunterladen und Extrahieren der JSON-Beispieldatei.

```
// Load data into the table.
public static void loadData(DynamoDbClient ddb, String tableName, String
fileName) throws IOException {
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(ddb)
        .build();

    DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    while (iter.hasNext()) {
        // Only add 200 Movies to the table.
        if (t == 200)
            break;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String info = currentNode.path("info").toString();

        Movies movies = new Movies();
        movies.setYear(year);
        movies.setTitle(title);
        movies.setInfo(info);

        // Put the data into the Amazon DynamoDB Movie table.
        mappedTable.putItem(movies);
        t++;
    }
}
```

Rufen Sie ein Element aus einer Tabelle ab.

```
public static void getItem(DynamoDbClient ddb) {
```

```
HashMap<String, AttributeValue> keyToGet = new HashMap<>();
keyToGet.put("year", AttributeValue.builder()
    .n("1933")
    .build());

keyToGet.put("title", AttributeValue.builder()
    .s("King Kong")
    .build());

GetItemRequest request = GetItemRequest.builder()
    .key(keyToGet)
    .tableName("Movies")
    .build();

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

    if (returnedItem != null) {
        Set<String> keys = returnedItem.keySet();
        System.out.println("Amazon DynamoDB table attributes: \n");

        for (String key1 : keys) {
            System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
        }
    } else {
        System.out.format("No item found with the key %s!\n", "year");
    }

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

### Vollständiges Beispiel.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* This Java example performs these tasks:
*
* 1. Creates the Amazon DynamoDB Movie table with partition and sort key.
* 2. Puts data into the Amazon DynamoDB table from a JSON document using the
* Enhanced client.
* 3. Gets data from the Movie table.
* 4. Adds a new item.
* 5. Updates an item.
* 6. Uses a Scan to query items using the Enhanced client.
* 7. Queries all items where the year is 2013 using the Enhanced Client.
* 8. Deletes the table.
*/

public class Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException {
        final String usage = ""

            Usage:
            <fileName>

            Where:
            fileName - The path to the moviedata.json file that you can
download from the Amazon DynamoDB Developer Guide.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = "Movies";
        String fileName = args[0];
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon DynamoDB example scenario.");
    }
}
```

```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(
    "1. Creating an Amazon DynamoDB table named Movies with a key named
year and a sort key named title.");
createTable(ddb, tableName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Loading data into the Amazon DynamoDB table.");
loadData(ddb, tableName, fileName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Getting data from the Movie table.");
getItem(ddb);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Putting a record into the Amazon DynamoDB table.");
putRecord(ddb);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Updating a record.");
updateTableItem(ddb, tableName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Scanning the Amazon DynamoDB table.");
scanMovies(ddb, tableName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Querying the Movies released in 2013.");
queryTable(ddb);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Deleting the Amazon DynamoDB table.");
deleteDynamoDBTable(ddb, tableName);
System.out.println(DASHES);
```

```
        ddb.close();
    }

    // Create a table with a Sort key.
    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE)
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .provisionedThroughput(ProvisionedThroughput.builder()
                .readCapacityUnits(10L)
                .writeCapacityUnits(10L)
                .build())
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();
    }
}
```

```
    try {
        CreateTableResponse response = ddb.createTable(request);
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        // Wait until the Amazon DynamoDB table is created.
        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        String newTable = response.tableDescription().tableName();
        System.out.println("The " + newTable + " was successfully created.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Query the table.
public static void queryTable(DynamoDbClient ddb) {
    try {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        QueryConditional queryConditional = QueryConditional
            .keyEqualTo(Key.builder()
                .partitionValue(2013)
                .build());

        // Get items in the table and write out the ID value.
        Iterator<Movies> results =
custTable.query(queryConditional).items().iterator();
        String result = "";

        while (results.hasNext()) {
            Movies rec = results.next();
            System.out.println("The title of the movie is " + rec.getTitle());
            System.out.println("The movie information is " + rec.getInfo());
        }
    }
}
```

```
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Scan the table.
public static void scanMovies(DynamoDbClient ddb, String tableName) {
    System.out.println("***** Scanning all movies.\n");
    try {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        Iterator<Movies> results = custTable.scan().items().iterator();
        while (results.hasNext()) {
            Movies rec = results.next();
            System.out.println("The movie title is " + rec.getTitle());
            System.out.println("The movie year is " + rec.getYear());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String tableName, String
fileName) throws IOException {
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(ddb)
        .build();

    DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
```

```
while (iter.hasNext()) {
    // Only add 200 Movies to the table.
    if (t == 200)
        break;
    currentNode = (ObjectNode) iter.next();

    int year = currentNode.path("year").asInt();
    String title = currentNode.path("title").asText();
    String info = currentNode.path("info").toString();

    Movies movies = new Movies();
    movies.setYear(year);
    movies.setTitle(title);
    movies.setInfo(info);

    // Put the data into the Amazon DynamoDB Movie table.
    mappedTable.putItem(movies);
    t++;
}

// Update the record to include show only directors.
public static void updateTableItem(DynamoDbClient ddb, String tableName) {
    HashMap<String, AttributeValue> itemKey = new HashMap<>();
    itemKey.put("year", AttributeValue.builder().n("1933").build());
    itemKey.put("title", AttributeValue.builder().s("King Kong").build());

    HashMap<String, AttributeValueUpdate> updatedValues = new HashMap<>();
    updatedValues.put("info", AttributeValueUpdate.builder()
        .value(AttributeValue.builder().s("{\"directors\": [\"Merian C.
Cooper\", \"Ernest B. Schoedsack\"]")
        .build())
        .action(AttributeAction.PUT)
        .build());

    UpdateItemRequest request = UpdateItemRequest.builder()
        .tableName(tableName)
        .key(itemKey)
        .attributeUpdates(updatedValues)
        .build();

    try {
        ddb.updateItem(request);
    } catch (ResourceNotFoundException e) {
```



```
        System.err.println(e.getMessage());
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    System.out.println("Item was updated!");
}

public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}

public static void putRecord(DynamoDbClient ddb) {
    try {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> table = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));

        // Populate the Table.
        Movies record = new Movies();
        record.setYear(2020);
        record.setTitle("My Movie2");
        record.setInfo("no info");
        table.putItem(record);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
    }
    System.out.println("Added a new movie to the table.");
}

public static void getItem(DynamoDbClient ddb) {

    HashMap<String, AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put("year", AttributeValue.builder()
        .n("1933")
        .build());

    keyToGet.put("title", AttributeValue.builder()
        .s("King Kong")
        .build());

    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName("Movies")
        .build();

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

        if (returnedItem != null) {
            Set<String> keys = returnedItem.keySet();
            System.out.println("Amazon DynamoDB table attributes: \n");

            for (String key1 : keys) {
                System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
            }
        } else {
            System.out.format("No item found with the key %s!\n", "year");
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.

- [BatchWriteItem](#)
- [CreateTable](#)
- [DeleteItem](#)
- [DeleteTable](#)
- [DescribeTable](#)
- [GetItem](#)
- [PutItem](#)
- [Abfrage](#)
- [Scan](#)
- [UpdateItem](#)

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 Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public class ScenarioPartiQLBatch {
    public static void main(String[] args) throws IOException {
        String tableName = "MoviesPartiQLBatch";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
```

```
        .build());

        System.out.println("***** Creating an Amazon DynamoDB table named
" + tableName
        + " with a key named year and a sort key named
title.");
        createTable(ddb, tableName);

        System.out.println("***** Adding multiple records into the " +
tableName
        + " table using a batch command.");
        putRecordBatch(ddb);

        System.out.println("***** Updating multiple records using a batch
command.");
        updateTableItemBatch(ddb);

        System.out.println("***** Deleting multiple records using a batch
command.");
        deleteItemBatch(ddb);

        System.out.println("***** Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new
ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
```

```
        .attributeName("year")
        .keyType(KeyType.HASH)
        .build();

    KeySchemaElement key2 = KeySchemaElement.builder()
        .attributeName("title")
        .keyType(KeyType.RANGE) // Sort
        .build();

    // Add KeySchemaElement objects to the list.
    tableKey.add(key);
    tableKey.add(key2);

    CreateTableRequest request = CreateTableRequest.builder()
        .keySchema(tableKey)

        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(new Long(10))
            .writeCapacityUnits(new Long(10))
            .build())
        .attributeDefinitions(attributeDefinitions)
        .tableName(tableName)
        .build();

    try {
        CreateTableResponse response = ddb.createTable(request);
        DescribeTableRequest tableRequest =
DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        // Wait until the Amazon DynamoDB table is created.
        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter
            .waitUntilTableExists(tableRequest);

        waiterResponse.matched().response().ifPresent(System.out::println);
        String newTable = response.tableDescription().tableName();
        System.out.println("The " + newTable + " was successfully
created.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
    }  
  }  
  
  public static void putRecordBatch(DynamoDbClient ddb) {  
    String sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE  
{'year':?, 'title' : ?, 'info' : ?}";  
    try {  
      // Create three movies to add to the Amazon DynamoDB table.  
      // Set data for Movie 1.  
      List<AttributeValue> parameters = new ArrayList<>();  
  
      AttributeValue att1 = AttributeValue.builder()  
        .n(String.valueOf("2022"))  
        .build();  
  
      AttributeValue att2 = AttributeValue.builder()  
        .s("My Movie 1")  
        .build();  
  
      AttributeValue att3 = AttributeValue.builder()  
        .s("No Information")  
        .build();  
  
      parameters.add(att1);  
      parameters.add(att2);  
      parameters.add(att3);  
  
      BatchStatementRequest statementRequestMovie1 =  
BatchStatementRequest.builder()  
        .statement(sqlStatement)  
        .parameters(parameters)  
        .build();  
  
      // Set data for Movie 2.  
      List<AttributeValue> parametersMovie2 = new ArrayList<>();  
      AttributeValue attMovie2 = AttributeValue.builder()  
        .n(String.valueOf("2022"))  
        .build();  
  
      AttributeValue attMovie2A = AttributeValue.builder()  
        .s("My Movie 2")  
        .build();  
  
      AttributeValue attMovie2B = AttributeValue.builder()
```

```
                .s("No Information")
                .build();

        parametersMovie2.add(attMovie2);
        parametersMovie2.add(attMovie2A);
        parametersMovie2.add(attMovie2B);

        BatchStatementRequest statementRequestMovie2 =
BatchStatementRequest.builder()

                .statement(sqlStatement)
                .parameters(parametersMovie2)
                .build();

        // Set data for Movie 3.
        List<AttributeValue> parametersMovie3 = new ArrayList<>();
        AttributeValue attMovie3 = AttributeValue.builder()
                .n(String.valueOf("2022"))
                .build();

        AttributeValue attMovie3A = AttributeValue.builder()
                .s("My Movie 3")
                .build();

        AttributeValue attMovie3B = AttributeValue.builder()
                .s("No Information")
                .build();

        parametersMovie3.add(attMovie3);
        parametersMovie3.add(attMovie3A);
        parametersMovie3.add(attMovie3B);

        BatchStatementRequest statementRequestMovie3 =
BatchStatementRequest.builder()

                .statement(sqlStatement)
                .parameters(parametersMovie3)
                .build();

        // Add all three movies to the list.
        List<BatchStatementRequest> myBatchStatementList = new
ArrayList<>();

        myBatchStatementList.add(statementRequestMovie1);
        myBatchStatementList.add(statementRequestMovie2);
        myBatchStatementList.add(statementRequestMovie3);
```

```
        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
                                .statements(myBatchStatementList)
                                .build();

        BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
        System.out.println("ExecuteStatement successful: " +
response.toString());
        System.out.println("Added new movies using a batch
command.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateTableItemBatch(DynamoDbClient ddb) {
    String sqlStatement = "UPDATE MoviesPartiQBatch SET info =
'directors\":[\"Merian C. Cooper\", \"Ernest B. Schoedsack' where year=? and
title=?";

    List<AttributeValue> parametersRec1 = new ArrayList<>();

    // Update three records.
    AttributeValue att1 = AttributeValue.builder()
        .n(String.valueOf("2022"))
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s("My Movie 1")
        .build();

    parametersRec1.add(att1);
    parametersRec1.add(att2);

    BatchStatementRequest statementRequestRec1 =
BatchStatementRequest.builder()
        .statement(sqlStatement)
        .parameters(parametersRec1)
        .build();

    // Update record 2.
    List<AttributeValue> parametersRec2 = new ArrayList<>();
```



```
AttributeValue attRec2 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec2a = AttributeValue.builder()
    .s("My Movie 2")
    .build();

parametersRec2.add(attRec2);
parametersRec2.add(attRec2a);
BatchStatementRequest statementRequestRec2 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec2)
    .build();

// Update record 3.
List<AttributeValue> parametersRec3 = new ArrayList<>();
AttributeValue attRec3 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec3a = AttributeValue.builder()
    .s("My Movie 3")
    .build();

parametersRec3.add(attRec3);
parametersRec3.add(attRec3a);
BatchStatementRequest statementRequestRec3 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec3)
    .build();

// Add all three movies to the list.
List<BatchStatementRequest> myBatchStatementList = new
ArrayList<>();
myBatchStatementList.add(statementRequestRec1);
myBatchStatementList.add(statementRequestRec2);
myBatchStatementList.add(statementRequestRec3);

BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
    .statements(myBatchStatementList)
```

```
        .build();

        try {
            BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
            System.out.println("ExecuteStatement successful: " +
response.toString());
            System.out.println("Updated three movies using a batch
command.");
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Item was updated!");
    }

    public static void deleteItemBatch(DynamoDbClient ddb) {
        String sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ?
and title=?";
        List<AttributeValue> parametersRec1 = new ArrayList<>();

        // Specify three records to delete.
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie 1")
            .build();

        parametersRec1.add(att1);
        parametersRec1.add(att2);

        BatchStatementRequest statementRequestRec1 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec1)
            .build();

        // Specify record 2.
        List<AttributeValue> parametersRec2 = new ArrayList<>();
        AttributeValue attRec2 = AttributeValue.builder()
            .n(String.valueOf("2022"))
```

```
        .build();

        AttributeValue attRec2a = AttributeValue.builder()
            .s("My Movie 2")
            .build();

        parametersRec2.add(attRec2);
        parametersRec2.add(attRec2a);
        BatchStatementRequest statementRequestRec2 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec2)
            .build();

        // Specify record 3.
        List<AttributeValue> parametersRec3 = new ArrayList<>();
        AttributeValue attRec3 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec3a = AttributeValue.builder()
            .s("My Movie 3")
            .build();

        parametersRec3.add(attRec3);
        parametersRec3.add(attRec3a);

        BatchStatementRequest statementRequestRec3 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec3)
            .build();

        // Add all three movies to the list.
        List<BatchStatementRequest> myBatchStatementList = new
ArrayList<>();
        myBatchStatementList.add(statementRequestRec1);
        myBatchStatementList.add(statementRequestRec2);
        myBatchStatementList.add(statementRequestRec3);

        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
            .statements(myBatchStatementList)
            .build();
```

```
        try {
            ddb.batchExecuteStatement(batchRequest);
            System.out.println("Deleted three movies using a batch
command.");
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName)
    {
        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }

    private static ExecuteStatementResponse
executeStatementRequest(DynamoDbClient ddb, String statement,
        List<AttributeValue> parameters) {
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
            .statement(statement)
            .parameters(parameters)
            .build();

        return ddb.executeStatement(request);
    }
}
```

- Weitere API-Informationen finden Sie unter [BatchExecuteStatement](#) in der APIAWS SDK for Java 2.x -Referenz für .

## 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 Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public class ScenarioPartiQ {
    public static void main(String[] args) throws IOException {
        final String usage = ""

            Usage:
                <fileName>

            Where:
                fileName - The path to the moviedata.json file that you can
download from the Amazon DynamoDB Developer Guide.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String fileName = args[0];
        String tableName = "MoviesPartiQ";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();
```

```
        System.out.println(
            "***** Creating an Amazon DynamoDB table named MoviesPartiQ with a
key named year and a sort key named title.");
        createTable(ddb, tableName);

        System.out.println("***** Loading data into the MoviesPartiQ table.");
        loadData(ddb, fileName);

        System.out.println("***** Getting data from the MoviesPartiQ table.");
        getItem(ddb);

        System.out.println("***** Putting a record into the MoviesPartiQ table.");
        putRecord(ddb);

        System.out.println("***** Updating a record.");
        updateTableItem(ddb);

        System.out.println("***** Querying the movies released in 2013.");
        queryTable(ddb);

        System.out.println("***** Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
```

```
        .keyType(KeyType.HASH)
        .build();

    KeySchemaElement key2 = KeySchemaElement.builder()
        .attributeName("title")
        .keyType(KeyType.RANGE) // Sort
        .build();

    // Add KeySchemaElement objects to the list.
    tableKey.add(key);
    tableKey.add(key2);

    CreateTableRequest request = CreateTableRequest.builder()
        .keySchema(tableKey)
        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(new Long(10))
            .writeCapacityUnits(new Long(10))
            .build())
        .attributeDefinitions(attributeDefinitions)
        .tableName(tableName)
        .build();

    try {
        CreateTableResponse response = ddb.createTable(request);
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        // Wait until the Amazon DynamoDB table is created.
        WaiterResponse<DescribeTableResponse> waiterResponse =
        dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        String newTable = response.tableDescription().tableName();
        System.out.println("The " + newTable + " was successfully created.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String fileName) throws
IOException {
```

```
String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
JsonParser parser = new JsonFactory().createParser(new File(fileName));
com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
Iterator<JsonNode> iter = rootNode.iterator();
ObjectNode currentNode;
int t = 0;
List<AttributeValue> parameters = new ArrayList<>();
while (iter.hasNext()) {

    // Add 200 movies to the table.
    if (t == 200)
        break;
    currentNode = (ObjectNode) iter.next();

    int year = currentNode.path("year").asInt();
    String title = currentNode.path("title").asText();
    String info = currentNode.path("info").toString();

    AttributeValue att1 = AttributeValue.builder()
        .n(String.valueOf(year))
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s(title)
        .build();

    AttributeValue att3 = AttributeValue.builder()
        .s(info)
        .build();

    parameters.add(att1);
    parameters.add(att2);
    parameters.add(att3);

    // Insert the movie into the Amazon DynamoDB table.
    executeStatementRequest(ddb, sqlStatement, parameters);
    System.out.println("Added Movie " + title);

    parameters.remove(att1);
    parameters.remove(att2);
    parameters.remove(att3);
```



```
        t++;
    }
}

public static void getItem(DynamoDbClient ddb) {

    String sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?";
    List<AttributeValue> parameters = new ArrayList<>();
    AttributeValue att1 = AttributeValue.builder()
        .n("2012")
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s("The Perks of Being a Wallflower")
        .build();

    parameters.add(att1);
    parameters.add(att2);

    try {
        ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
        System.out.println("ExecuteStatement successful: " +
response.toString());

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void putRecord(DynamoDbClient ddb) {

    String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
    try {
        List<AttributeValue> parameters = new ArrayList<>();

        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2020"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie")
```

```
        .build();

        AttributeValue att3 = AttributeValue.builder()
            .s("No Information")
            .build();

        parameters.add(att1);
        parameters.add(att2);
        parameters.add(att3);

        executeStatementRequest(ddb, sqlStatement, parameters);
        System.out.println("Added new movie.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateTableItem(DynamoDbClient ddb) {

    String sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian
C. Cooper\", \"Ernest B. Schoedsack' where year=? and title=?";
    List<AttributeValue> parameters = new ArrayList<>();
    AttributeValue att1 = AttributeValue.builder()
        .n(String.valueOf("2013"))
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s("The East")
        .build();

    parameters.add(att1);
    parameters.add(att2);

    try {
        executeStatementRequest(ddb, sqlStatement, parameters);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Item was updated!");
}
```

```
// Query the table where the year is 2013.
public static void queryTable(DynamoDbClient ddb) {
    String sqlStatement = "SELECT * FROM MoviesPartiQ where year = ? ORDER BY
year";
    try {

        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2013"))
            .build();
        parameters.add(att1);

        // Get items in the table and write out the ID value.
        ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
        System.out.println("ExecuteStatement successful: " +
response.toString());

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}

private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient
ddb, String statement,
    List<AttributeValue> parameters) {
```

```
ExecuteStatementRequest request = ExecuteStatementRequest.builder()
    .statement(statement)
    .parameters(parameters)
    .build();

return ddb.executeStatement(request);
}

private static void processResults(ExecuteStatementResponse
executeStatementResult) {
    System.out.println("ExecuteStatement successful: " +
executeStatementResult.toString());
}
}
```

- Weitere API-Informationen finden Sie unter [ExecuteStatement](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Amazon EC2-Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon EC2 verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Erste Schritte

#### Hello Amazon EC2

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon EC2.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupIds(groupId)
            .build();

        // Use a paginator.
        DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
        listGroups.stream()
            .flatMap(r -> r.securityGroups().stream())
            .forEach(group -> System.out
                .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeSecurityGroups](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Themen

- [Aktionen](#)
- [Szenarien](#)

## Aktionen

### Zuweisen einer Elastic-IP-Adresse

Das folgende Codebeispiel zeigt, wie Sie eine Elastic IP-Adresse für Amazon EC2 zuweisen.

SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String allocateAddress(Ec2Client ec2) {
    try {
        AllocateAddressRequest allocateRequest =
AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        AllocateAddressResponse allocateResponse =
ec2.allocateAddress(allocateRequest);
        return allocateResponse.allocationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Weitere API-Informationen finden Sie unter [AllocateAddress](#) in der APIAWS SDK for Java 2.x - Referenz für .

### Zuordnen einer Elastic-IP-Adresse zu einer Instance

Das folgende Codebeispiel zeigt, wie Sie eine Elastic IP-Adresse einer Amazon EC2-Instance zuordnen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String associateAddress(Ec2Client ec2, String instanceId, String
allocationId) {
    try {
        AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
            .instanceId(instanceId)
            .allocationId(allocationId)
            .build();

        AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
        return associateResponse.associationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Weitere API-Informationen finden Sie unter [AssociateAddress](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Eine Sicherheitsgruppe erstellen

Das folgende Codebeispiel zeigt, wie Sie eine Amazon EC2Sicherheitsgruppe erstellen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
    String myIpAddress) {
    try {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
        IpRange ipRange = IpRange.builder()
            .cidrIp(myIpAddress + "/0")
            .build();

        IpPermission ipPerm = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(80)
            .fromPort(80)
            .ipRanges(ipRange)
            .build();

        IpPermission ipPerm2 = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(22)
            .fromPort(22)
            .ipRanges(ipRange)
            .build();

        AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(groupName)
```



```
        .ipPermissions(ipPerm, ipPerm2)
        .build();

        ec2.authorizeSecurityGroupIngress(authRequest);
        System.out.println("Successfully added ingress policy to security group
" + groupName);
        return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Weitere API-Informationen finden Sie unter [CreateSecurityGroup](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erstellen eines Sicherheitsschlüsselpaares

Das folgende Codebeispiel zeigt, wie Sie ein Sicherheitsschlüsselpaar für Amazon EC2 erstellen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void createKeyPair(Ec2Client ec2, String keyName, String fileName)
{
    try {
        CreateKeyPairRequest request = CreateKeyPairRequest.builder()
            .keyName(keyName)
            .build();

        CreateKeyPairResponse response = ec2.createKeyPair(request);
        String content = response.keyMaterial();
        BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
        writer.write(content);
    }
}
```

```
        writer.close();
        System.out.println("Successfully created key pair named " + keyName);

    } catch (Ec2Exception | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [CreateKeyPair](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen und Ausführen einer Instance

Das folgende Codebeispiel zeigt, wie Sie eine Amazon EC2 erstellen und ausführen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.InstanceType;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
* This code example requires an AMI value. You can learn more about this value
* by reading this documentation topic:
*
* https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/AMIs.html
*/
public class CreateInstance {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <name> <amiId>

            Where:
                name - An instance name value that you can obtain from the AWS
Console (for example, ami-xxxxxx5c8b987b1a0).\s
                amiId - An Amazon Machine Image (AMI) value that you can obtain
from the AWS Console (for example, i-xxxxxx2734106d0ab).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        String amiId = args[1];
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .build();

        String instanceId = createEC2Instance(ec2, name, amiId);
        System.out.println("The Amazon EC2 Instance ID is " + instanceId);
        ec2.close();
    }

    public static String createEC2Instance(Ec2Client ec2, String name, String amiId)
    {
        RunInstancesRequest runRequest = RunInstancesRequest.builder()
            .imageId(amiId)
            .instanceType(InstanceType.T1_MICRO)
            .maxCount(1)
            .minCount(1)
            .build();
    }
}
```

```
// Use a waiter to wait until the instance is running.
System.out.println("Going to start an EC2 instance using a waiter");
RunInstancesResponse response = ec2.runInstances(runRequest);
String instanceIdVal = response.instances().get(0).instanceId();
ec2.waiter().waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal));
Tag tag = Tag.builder()
    .key("Name")
    .value(name)
    .build();

CreateTagsRequest tagRequest = CreateTagsRequest.builder()
    .resources(instanceIdVal)
    .tags(tag)
    .build();

try {
    ec2.createTags(tagRequest);
    System.out.printf("Successfully started EC2 Instance %s based on AMI
%s", instanceIdVal, amiId);
    return instanceIdVal;

} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

return "";
}
}
```

- Weitere API-Informationen finden Sie unter [RunInstances](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen einer Sicherheitsgruppe

Das folgende Codebeispiel zeigt, wie Sie eine Amazon EC2Sicherheitsgruppe löschen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {
    try {
        DeleteSecurityGroupRequest request =
DeleteSecurityGroupRequest.builder()
        .groupId(groupId)
        .build();

        ec2.deleteSecurityGroup(request);
        System.out.println("Successfully deleted security group with Id " +
groupId);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteSecurityGroup](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Löschen eines Sicherheitsschlüsselpaars

Das folgende Codebeispiel zeigt, wie Sie ein Amazon EC2-Sicherheitsschlüsselpaar löschen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void deleteKeys(Ec2Client ec2, String keyPair) {
    try {
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        ec2.deleteKeyPair(request);
        System.out.println("Successfully deleted key pair named " + keyPair);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteKeyPair](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beschreiben von Instances

Das folgende Codebeispiel zeigt, wie Amazon-EC2 beschrieben werden.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.paginators.DescribeInstancesIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DescribeInstances {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .build();

        describeEC2Instances(ec2);
        ec2.close();
    }

    public static void describeEC2Instances(Ec2Client ec2) {
        try {
            DescribeInstancesRequest request = DescribeInstancesRequest.builder()
                .maxResults(10)
                .build();

            DescribeInstancesIterable instancesIterable =
ec2.describeInstancesPaginator(request);
            instancesIterable.stream()
                .flatMap(r -> r.reservations().stream())
                .flatMap(reservation -> reservation.instances().stream())
                .forEach(instance -> {
                    System.out.println("Instance Id is " + instance.instanceId());
                    System.out.println("Image id is " + instance.imageId());
                    System.out.println("Instance type is " +
instance.instanceType());
                    System.out.println("Instance state name is " +
instance.state().name());
                    System.out.println("Monitoring information is " +
instance.monitoring().state());
                });

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorCode());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeInstances](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Aufhebung der Zuordnung einer Elastic IP-Adresse zu einer Instance

Das folgende Codebeispiel zeigt, wie Sie eine Elastic IP-Adresse von einer Amazon EC2-Instance trennen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void disassociateAddress(Ec2Client ec2, String associationId) {
    try {
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();

        ec2.disassociateAddress(addressRequest);
        System.out.println("You successfully disassociated the address!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DisassociateAddress](#) in der APIAWS SDK for Java 2.x -Referenz für .



## Abrufen von Daten zu einer Sicherheitsgruppe

Das folgende Codebeispiel zeigt, wie Sie Daten zu einer Amazon EC2Sicherheitsgruppe abrufen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupIds(groupId)
            .build();

        // Use a paginator.
        DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
        listGroups.stream()
            .flatMap(r -> r.securityGroups().stream())
            .forEach(group -> System.out
                .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeSecurityGroups](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Abrufen von Daten über Instance-Typen

Das folgende Codebeispiel zeigt, wie Sie Daten zu Amazon EC2-Typen abrufen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
// Get a list of instance types.
public static String getInstanceTypes(Ec2Client ec2) {
    String instanceType;
    try {
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
        .maxResults(10)
        .build();

        DescribeInstanceTypesResponse response =
ec2.describeInstanceTypes(typesRequest);
        List<InstanceTypeInfo> instanceTypes = response.instanceTypes();
        for (InstanceTypeInfo type : instanceTypes) {
            System.out.println("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
            System.out.println("Network information is " +
type.networkInfo().toString());
            System.out.println("Instance type is " +
type.instanceType().toString());
            instanceType = type.instanceType().toString();
            if (instanceType.compareTo("t2.2xlarge") == 0){
                return instanceType;
            }
        }
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- Weitere API-Informationen finden Sie unter [DescribeInstanceTypes](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Sicherheitsschlüsselpaare auflisten

Das folgende Codebeispiel zeigt, wie Amazon EC2-Sicherheitsschlüsselpaare auflisten.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void describeKeys(Ec2Client ec2) {
    try {
        DescribeKeyPairsResponse response = ec2.describeKeyPairs();
        response.keyPairs().forEach(keyPair -> System.out.printf(
            "Found key pair with name %s " +
            "and fingerprint %s",
            keyPair.keyName(),
            keyPair.keyFingerprint()));
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeKeyPairs](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Freigeben einer Elastic-IP-Adresse

Das folgende Codebeispiel zeigt, wie Sie eine Elastic IP-Adresse freigeben.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void releaseEC2Address(Ec2Client ec2, String allocId) {
    try {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();


        ec2.releaseAddress(request);
        System.out.println("Successfully released Elastic IP address " +
allocId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [ReleaseAddress](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Eingehende Regeln für eine Sicherheitsgruppe festlegen

Das folgende Codebeispiel zeigt, wie eingehende Regeln für eine Amazon EC2Sicherheitsgruppe festgelegt werden.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
    String myIpAddress) {
    try {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
        IpRange ipRange = IpRange.builder()
            .cidrIp(myIpAddress + "/0")
            .build();

        IpPermission ipPerm = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(80)
            .fromPort(80)
            .ipRanges(ipRange)
            .build();

        IpPermission ipPerm2 = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(22)
            .fromPort(22)
            .ipRanges(ipRange)
            .build();

        AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(groupName)
            .ipPermissions(ipPerm, ipPerm2)
            .build();

        ec2.authorizeSecurityGroupIngress(authRequest);
        System.out.println("Successfully added ingress policy to security group
" + groupName);
        return resp.groupId();

    } catch (Ec2Exception e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Weitere API-Informationen finden Sie unter [AuthorizeSecurityGroupIngress](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Starten einer Instance

Das folgende Codebeispiel zeigt, wie Sie eine Amazon EC2 starten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void startInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();

    StartInstancesRequest request = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to run. This
will take a few minutes.");
    ec2.startInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceRunning(instanceRequest);
}
```

```

        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully started instance " + instanceId);
    }

```

- Weitere API-Informationen finden Sie unter [StartInstances](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Anhalten einer Instance

Das folgende Codebeispiel zeigt, wie Sie eine Amazon EC2 anhalten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

public static void stopInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();

    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
will take a few minutes.");
    ec2.stopInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully stopped instance " + instanceId);
}

```

- Weitere API-Informationen finden Sie unter [StopInstances](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beenden einer Instance

Das folgende Codebeispiel zeigt, wie Sie eine Amazon EC2 beenden.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void terminateEC2(Ec2Client ec2, String instanceId) {
    try {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
            .overrideConfiguration(b -> b.maxAttempts(100))
            .client(ec2)
            .build();

        TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to
terminate. This will take a few minutes.");
        ec2.terminateInstances(ti);
        DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse = ec2Waiter
            .waitUntilInstanceTerminated(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully started instance " + instanceId);
        System.out.println(instanceId + " is terminated!");
    }
}
```



```
    } catch (Ec2Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Weitere API-Informationen finden Sie unter [TerminateInstances](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Szenarien

### Erstellen und Verwalten eines ausfallsicheren Services

Das folgende Codebeispiel zeigt, wie Sie einen Webservice mit Load Balancing erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Antwort des Webserver auf Anfragen und Zustandsprüfungen, indem Sie AWS Systems Manager die Parameter aktualisieren.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\\
\\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\\
\\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\\
\\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\\
\\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\\0", "-");

    public static void main(String[] args) throws IOException, InterruptedException
    {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and Manage
a Resilient Service!");
    }
}
```

```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("A - SETUP THE RESOURCES");
System.out.println("Press Enter when you're ready to start deploying
resources.");
in.nextLine();
deploy(loadBalancer);
System.out.println(DASHES);
System.out.println(DASHES);
System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
System.out.println("Press Enter when you're ready.");
in.nextLine();
demo(loadBalancer);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("C - DELETE THE RESOURCES");
System.out.println("""
    This concludes the demo of how to build and manage a resilient
service.
    To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
    that were created for this demo.
    """);

System.out.println("\n Do you want to delete the resources (y/n)? ");
String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

if (userInput.equals("y")) {
    // Delete resources here
    deleteResources(loadBalancer, autoScaler, database);
    System.out.println("Resources deleted.");
} else {
    System.out.println("""
        Okay, we'll leave the resources intact.
        Don't forget to delete them when you're done with them or you
might incur unexpected charges.
    """);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The example has completed. ");
```

```

        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
        throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
            """
                For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources
                to set up a load-balanced web service endpoint and explore
some ways to make it resilient
                against various kinds of failures.

                Some of the resources create by this demo are:
                \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.
                \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
            """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);
    }

```

```
System.out.println(DASHES);
System.out.println("Creating and populating a DynamoDB table named " +
tableName);
Database database = new Database();
database.createTable(tableName, fileName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("""
    Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
    This script starts a Python web server defined in the `server.py`
script. The web server
    listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
    For demo purposes, this server is run as the root user. In
production, the best practice is to
    run a web server, such as Apache, with least-privileged credentials.

    The template also defines an IAM policy that each instance uses to
assume a role that grants
    permissions to access the DynamoDB recommendation table and Systems
Manager parameters
    that control the flow of the demo.
    """);

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(
    "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
System.out.println("*** Wait 30 secs for the VPC to be created");
TimeUnit.SECONDS.sleep(30);
AutoScaler autoScaler = new AutoScaler();
String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

System.out.println("""
    At this point, you have EC2 instances created. Once each instance
starts, it listens for
```

```
        HTTP requests. You can see these instances in the console or
continue with the demo.
        Press Enter when you're ready to continue.
        """);

in.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Creating variables that control the flow of the demo.");
ParameterHelper paramHelper = new ParameterHelper();
paramHelper.reset();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("""
    Creating an Elastic Load Balancing target group and load balancer.
The target group
    defines how the load balancer connects to instances. The load
balancer provides a
    single endpoint where clients connect and dispatches requests to
instances in the group.
    """);

String vpcId = autoScaler.getDefaultVPC();
List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
System.out.println("Verifying access to the load balancer endpoint...");
boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
if (!wasSuccessful) {
    System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to "http://checkip.amazonaws.com"
    HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
    try {
```

```

        // Execute the request and get the response
        HttpResponse response = httpClient.execute(httpGet);

        // Read the response content.
        String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

        // Print the public IP address.
        System.out.println("Public IP Address: " + ipAddress);
        GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
        if (!groupInfo.isPortOpen()) {
            System.out.println("""
                For this example to work, the default security group for
your default VPC must
                allow access from this computer. You can either add it
automatically from this
                example or add it yourself using the AWS Management
Console.
                """);

            System.out.println(
                "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
            System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
            String ans = in.nextLine();
            if ("y".equalsIgnoreCase(ans)) {
                autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                System.out.println("Security group rule added.");
            } else {
                System.out.println("No security group rule added.");
            }
        }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
    } else if (wasSuccessful) {
        System.out.println("Your load balancer is ready. You can access it by
browsing to:");
        System.out.println("\t http://" + elbDnsName);
    } else {

```

```
        System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
        System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
        System.out.println("you can successfully make a GET request to the load
balancer.");
    }

    System.out.println("Press Enter when you're ready to continue with the
demo.");
    in.nextLine();
}

// A method that controls the demo part of the Java program.
public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    ParameterHelper paramHelper = new ParameterHelper();
    System.out.println("Read the ssm_only_policy.json file");
    String ssmOnlyPolicy = readFileAsString(ssmJSON);

    System.out.println("Resetting parameters to starting values for demo.");
    paramHelper.reset();

    System.out.println(
        """
                This part of the demonstration shows how to toggle
different parts of the system
                to create situations where the web service fails, and shows
how using a resilient
                architecture can keep the web service running in spite of
these failures.

                At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
        """);
    demoChoices(loadBalancer);

    System.out.println(
        """
                The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                The table name is contained in a Systems Manager parameter
named self.param_helper.table.
        """);
}
```



```
        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
        """);
    paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

    System.out.println(
        ""
        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
        """);
    demoChoices(loadBalancer);

    System.out.println(
        ""
        Instead of failing when the recommendation service fails,
the web service can return a static response.
        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
        """);
    paramHelper.put(paramHelper.failureResponse, "static");

    System.out.println("""
        Now, sending a GET request to the load balancer endpoint returns a
static response.
        The service still reports as healthy because health checks are still
shallow.
        """);
    demoChoices(loadBalancer);

    System.out.println("Let's reinstate the recommendation service.");
    paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

    System.out.println("""
        Let's also substitute bad credentials for one of the instances in
the target group so that it can't
        access the DynamoDB recommendation table. We will get an instance id
value.
        """);

    LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
    AutoScaler autoScaler = new AutoScaler();
```

```
// Create a new instance profile based on badCredsProfileName.
templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
System.out.println("The association Id value is " + profileAssociationId);
System.out.println("Replacing the profile for instance " + badInstanceId
+ " with a profile that contains bad credentials");
autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

System.out.println(
    """
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
        depending on which instance is selected by the load
balancer.
    """);

demoChoices(loadBalancer);

System.out.println("""
    Let's implement a deep health check. For this demo, a deep health
check tests whether
    the web service can access the DynamoDB table that it depends on for
recommendations. Note that
    the deep health check is only for ELB routing and not for Auto
Scaling instance health.
    This kind of deep health check is not recommended for Auto Scaling
instance health, because it
    risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
    """);

System.out.println("""
    By implementing deep health checks, the load balancer can detect
when one of the instances is failing
    and take that instance out of rotation.
    """);

paramHelper.put(paramHelper.healthCheck, "deep");
```

```
        System.out.println("""
            Now, checking target health indicates that the instance with bad
credentials
            is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
            instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
            the load balancer takes unhealthy instances out of its rotation.
            """);

        demoChoices(loadBalancer);

        System.out.println(
            """
                Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                instance is to terminate it and let the auto scaler start a
new instance to replace it.
                """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
            Even while the instance is terminating and the new instance is
starting, sending a GET
            request to the web service continues to get a successful
recommendation response because
            the load balancer routes requests to the healthy instances. After
the replacement instance
            starts and reports as healthy, it is included in the load balancing
rotation.

            Note that terminating and replacing an instance typically takes
several minutes, during which time you
            can see the changing health check status until the new instance is
running and healthy.
            """);

        demoChoices(loadBalancer);
        System.out.println(
            "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
```

```
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
    InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode = response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
```

```

        new
        InputStreamReader(response.getEntity().getContent()));
        StringBuilder jsonResponse = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            jsonResponse.append(line);
        }
        reader.close();

        // Print the formatted JSON response.
        System.out.println("Full Response:\n");
        System.out.println(jsonResponse.toString());

        // Close the HTTP client.
        httpClient.close();

    }
    case 1 -> {
        System.out.println("\nChecking the health of load balancer
targets:\n");

        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                                target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println("""
Note that it can take a minute or two for the health
check to update
                                after changes are made.
                                """);
    }
    case 2 -> {
        System.out.println("\nOkay, let's move on.");
        System.out.println("-".repeat(88));
        return; // Exit the method when choice is 2
    }
    default -> System.out.println("You must choose a value between
0-2. Please select again.");
}

} catch (java.util.InputMismatchException e) {

```

```
        System.out.println("Invalid input. Please select again.");
        scanner.nextLine(); // Clear the input buffer.
    }
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
```

```
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance is
 * terminated, it can no longer be accessed.
 */
public void terminateInstance(String instanceId) {
    TerminateInstanceInAutoScalingGroupRequest terminateInstanceRequest =
    TerminateInstanceInAutoScalingGroupRequest
        .builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

    getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceRequest);
    System.out.format("Terminated instance %s.", instanceId);
}

/**
 * Replaces the profile associated with a running instance. After the profile is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
 * When
 * the instance is ready, Systems Manager is used to restart the Python web
 * server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
    throws InterruptedException {
    // Create an IAM instance profile specification.
```

```
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
    .builder()
    .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
                                // name.
    .build();

// Replace the IAM instance profile association for the EC2 instance.
ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
    .builder()
    .iamInstanceProfile(iamInstanceProfile)
    .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
    .build();

try {
    getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
    // Handle the response as needed.
} catch (Ec2Exception e) {
    // Handle exceptions, log, or report the error.
    System.err.println("Error: " + e.getMessage());
}

System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
    newInstanceProfileName);
TimeUnit.SECONDS.sleep(15);
boolean instReady = false;
int tries = 0;

// Reboot after 60 seconds
while (!instReady) {
    if (tries % 6 == 0) {
        getEc2Client().rebootInstances(RebootInstancesRequest.builder()
            .instanceIds(instanceId)
            .build());
        System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
    }
    tries++;
    try {
        TimeUnit.SECONDS.sleep(10);
    }
}
```



```

    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
    List<InstanceInformation> instanceInformationList =
informationResponse.getInstanceInformationList();
    for (InstanceInformation info : instanceInformationList) {
        if (info.getInstanceId().equals(instanceId)) {
            instReady = true;
            break;
        }
    }
}

SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
    .instanceIds(instanceId)
    .documentName("AWS-RunShellScript")
    .parameters(Collections.singletonMap("commands",
80")))
        Collections.singletonList("cd / && sudo python3 server.py

    .build();

getSSMClient().sendCommand(sendCommandRequest);
System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
}

public void openInboundPort(String secGroupId, String port, String ipAddress) {
    AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
        .groupName(secGroupId)
        .cidrIp(ipAddress)
        .fromPort(Integer.parseInt(port))
        .build();

    getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
    System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
}

/**
 * Detaches a role from an instance profile, detaches policies from the role,

```

```
    * and deletes all the resources.
    */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
            getInstanceProfileRequest =
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                .builder()
                .instanceProfileName(profileName)
                .build();

            GetInstanceProfileResponse response =
            getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
            RemoveRoleFromInstanceProfileRequest.builder()
                .instanceProfileName(profileName)
                .roleName(roleName)
                .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
            DeleteInstanceProfileRequest.builder()
                .instanceProfileName(profileName)
                .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                .roleName(roleName)
                .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
            rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(attachedPolicy.policyArn())
```

```
        .build();

        getIAMClient().detachRolePolicy(request);
        System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
    }

    getIAMClient().deleteRole(deleteRoleRequest);
    System.out.println("Instance profile and role deleted.");

} catch (IamException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
    System.out.format(templateName + " was deleted.");
}

public void deleteAutoScaleGroup(String groupName) {
    DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
        .autoScalingGroupName(groupName)
        .forceDelete(true)
        .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println(groupName + " was deleted.");
}

/*
 * Verify the default security group of the specified VPC allows ingress from
 * this
 * computer. This can be done by allowing ingress from this computer's IP
 * address. In some situations, such as connecting from a corporate network, you
 * must instead specify a prefix list ID. You can also temporarily open the port
 * to
 * any IP address while running this example. If you do, be sure to remove
 * public
 * access when you're done.
 */
```

```
    *
    */
public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
    boolean portIsOpen = false;
    GroupInfo groupInfo = new GroupInfo();
    try {
        Filter filter = Filter.builder()
            .name("group-name")
            .values("default")
            .build();

        Filter filter1 = Filter.builder()
            .name("vpc-id")
            .values(VPC)
            .build();

        DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
            .filters(filter, filter1)
            .build();

        DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
            .describeSecurityGroups(securityGroupsRequest);
        String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
        groupInfo.setGroupName(securityGroup);

        for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
            System.out.println("Found security group: " + secGroup.groupId());

            for (IpPermission ipPermission : secGroup.ipPermissions()) {
                if (ipPermission.fromPort() == port) {
                    System.out.println("Found inbound rule: " + ipPermission);
                    for (IpRange ipRange : ipPermission.ipRanges()) {
                        String cidrIp = ipRange.cidrIp();
                        if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                            System.out.println(cidrIp + " is applicable");
                            portIsOpen = true;
                        }
                    }
                }

                if (!ipPermission.prefixListIds().isEmpty()) {
                    System.out.println("Prefix lList is applicable");
                }
            }
        }
    }
}
```

```
        portIsOpen = true;
    }

    if (!portIsOpen) {
        System.out
            .println("The inbound rule does not appear to be
open to either this computer's IP,"
                    + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
    } else {
        break;
    }
}
}

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}

groupInfo.setPortOpen(portIsOpen);
return groupInfo;
}

/**
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {
        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
            .autoScalingGroupName(asGroupName)
            .targetGroupARNs(targetGroupARN)
            .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
        System.out.println("Attached load balancer to " + asGroupName);
    }
}
```

```
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

    // Get availability zones.
    software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
    .builder()
    .build();

    DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
    List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
    .collect(Collectors.toList());

    String availabilityZones = String.join(",", availabilityZoneNames);
    LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
    .launchTemplateName(templateName)
    .version("$Default")
    .build();

    String[] zones = availabilityZones.split(",");
    CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
    .launchTemplate(specification)
    .availabilityZones(zones)
    .maxSize(groupSize)
    .minSize(groupSize)
    .autoScalingGroupName(autoScalingGroupName)
    .build();

    try {
        getAutoScalingClient().createAutoScalingGroup(groupRequest);
    }
```

```
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
    return zones;
}

public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();

    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
        .builder()
        .filters(defaultFilter)
        .build();

    DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
    return response.vpcs().get(0).vpcId();
}

// Gets the default subnets in a VPC for a specified list of Availability Zones.
public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
    List<Subnet> subnets = null;
    Filter vpcFilter = Filter.builder()
        .name("vpc-id")
        .values(vpcId)
        .build();

    Filter azFilter = Filter.builder()
        .name("availability-zone")
        .values(availabilityZones)
        .build();

    Filter defaultForAZ = Filter.builder()
        .name("default-for-az")
        .values("true")
        .build();
```

```
DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
    .filters(vpcFilter, azFilter, defaultForAZ)
    .build();

DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
subnets = response.subnets();
return subnets;
}

// Gets data about the instances in the EC2 Auto Scaling group.
public String getBadInstance(String groupName) {
    DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
    .autoScalingGroupNames(groupName)
    .build();

    DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
    AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
    List<String> instanceIds = autoScalingGroup.instances().stream()
        .map(instance -> instance.instanceId())
        .collect(Collectors.toList());

    String[] instanceIdArray = instanceIds.toArray(new String[0]);
    for (String instanceId : instanceIdArray) {
        System.out.println("Instance ID: " + instanceId);
        return instanceId;
    }
    return "";
}

// Gets data about the profile associated with an instance.
public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();
```



```
DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
    .describeIamInstanceProfileAssociations(associationsRequest);
return response.iamInstanceProfileAssociations().get(0).associationId();
}

public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
    ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
    ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
    for (Policy policy : listPoliciesResponse.policies()) {
        if (policy.policyName().equals(policyName)) {
            // List the entities (users, groups, roles) that are attached to the
policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
    .builder()
    .policyArn(policy.arn())
    .build();
            ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
    .listEntitiesForPolicy(listEntitiesRequest);
            if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                || !listEntitiesResponse.policyRoles().isEmpty()) {
                // Detach the policy from any entities it is attached to.
                DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
    .policyArn(policy.arn())
    .roleName(roleName) // Specify the name of the IAM role
    .build();

                getIAMClient().detachRolePolicy(detachPolicyRequest);
                System.out.println("Policy detached from entities.");
            }

            // Now, you can delete the policy.
            DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
    .policyArn(policy.arn())
    .build();
```

```
        getIAMClient().deletePolicy(deletePolicyRequest);
        System.out.println("Policy deleted successfully.");
        break;
    }
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
    RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(InstanceProfile)
        .roleName(roleName) // Remove the extra dot here
        .build();

    getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
    System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
}

// Delete the instance profile after removing all roles
DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
    .instanceProfileName(InstanceProfile)
    .build();

getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
System.out.println(InstanceProfile + " Deleted");
System.out.println("All roles and policies are deleted.");
}
}
```

Erstellen Sie eine Klasse, die Elastic-Load-Balancing-Aktionen beinhaltet.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
            .names(targetGroupName)
            .build();

        DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
            .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
            .build();

        DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
```

```

    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {

```

```
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to the ELB.
    HttpGet httpGet = new HttpGet("http://" + elbDnsName);
    try {
        while ((!success) && (retries > 0)) {
            // Execute the request and get the response.
            HttpResponse response = httpClient.execute(httpGet);
            int statusCode = response.getStatusLine().getStatusCode();
            System.out.println("HTTP Status Code: " + statusCode);
            if (statusCode == 200) {
                success = true;
            } else {
                retries--;
                System.out.println("Got connection error from load balancer
endpoint, retrying...");
                TimeUnit.SECONDS.sleep(15);
            }
        }

        } catch (org.apache.http.conn.HttpHostConnectException e) {
            System.out.println(e.getMessage());
        }

        System.out.println("Status.." + success);
        return success;
    }

    /**
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
            .healthCheckPath("/healthcheck")
            .healthCheckTimeoutSeconds(5)
            .port(port)
            .vpcId(vpcId)
```

```
        .name(targetGroupName)
        .protocol(protocol)
        .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /**
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
        String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                .map(Subnet::subnetId)
                .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                .subnets(subnetIdStrings)
                .name(lbName)
                .scheme("internet-facing")
                .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
```

```
        .loadBalancerArns(lbARN)
        .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " + e.getMessage());
        }
        return false;
    }

    /**
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended, such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
```



```
* MediaType,
* forms a unique identifier for the recommended item.
*/
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("ItemId")
                    .attributeType(ScalarAttributeType.N)
                    .build())
            .keySchema(
                KeySchemaElement.builder()
                    .attributeName("MediaType")
                    .keyType(KeyType.HASH)
                    .build(),
                KeySchemaElement.builder()
                    .attributeName("ItemId")
                    .keyType(KeyType.RANGE)
                    .build())
            .provisionedThroughput(
                ProvisionedThroughput.builder()
                    .readCapacityUnits(5L)
                    .writeCapacityUnits(5L)
                    .build())
            .build();

        getDynamoDbClient().createTable(createTableRequest);
        System.out.println("Creating table " + tableName + "...");

        // Wait until the Amazon DynamoDB table is created.
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();
    }
}
```

```
        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Table " + tableName + " created.");

        // Add records to the table.
        populateTable(fileName, tableName);
    }
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.
public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
}
```

```
        System.out.println("Added all records to the " + tableName);
    }
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
            .overwrite(true)
            .type("String")
            .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [AttachLoadBalancerTargetGroups](#)
  - [CreateAutoScalingGroup](#)
  - [CreateInstanceProfile](#)

- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacelamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

## Erste Schritte mit Instances

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 Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Creates an RSA key pair and saves the private key data as a .pem file.
 * 2. Lists key pairs.
 * 3. Creates a security group for the default VPC.
 * 4. Displays security group information.
 * 5. Gets a list of Amazon Linux 2 AMIs and selects one.
 * 6. Gets more information about the image.
 * 7. Gets a list of instance types that are compatible with the selected AMI's
 * architecture.
 * 8. Creates an instance with the key pair, security group, AMI, and an
 * instance type.
 * 9. Displays information about the instance.
 * 10. Stops the instance and waits for it to stop.
 * 11. Starts the instance and waits for it to start.
 * 12. Allocates an Elastic IP address and associates it with the instance.
 * 13. Displays SSH connection info for the instance.
 * 14. Disassociates and deletes the Elastic IP address.
 * 15. Terminates the instance and waits for it to terminate.
 * 16. Deletes the security group.
 * 17. Deletes the key pair.
 */
```

```
public class EC2Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {

        final String usage = ""

            Usage:
                <keyName> <fileName> <groupName> <groupDesc> <vpcId>

            Where:
                keyName - A key pair name (for example, TestKeyPair).\s
                fileName - A file name where the key information is written to.
\s
                groupName - The name of the security group.\s
                groupDesc - The description of the security group.\s
                vpcId - A VPC Id value. You can get this value from the AWS
Management Console.\s
                myIpAddress - The IP address of your development machine.\s

            """;

        if (args.length != 6) {
            System.out.println(usage);
            System.exit(1);
        }

        String keyName = args[0];
        String fileName = args[1];
        String groupName = args[2];
        String groupDesc = args[3];
        String vpcId = args[4];
        String myIpAddress = args[5];

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

        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();

        System.out.println(DASHES);
    }
}
```

```
System.out.println("Welcome to the Amazon EC2 example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create an RSA key pair and save the private key
material as a .pem file.");
createKeyPair(ec2, keyName, fileName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. List key pairs.");
describeKeys(ec2);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Create a security group.");
String groupId = createSecurityGroup(ec2, groupName, groupDesc, vpcId,
myIpAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Display security group info for the newly created
security group.");
describeSecurityGroups(ec2, groupId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Get a list of Amazon Linux 2 AMIs and selects one
with amzn2 in the name.");
String instanceId = getParaValues(ssmClient);
System.out.println("The instance Id is " + instanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Get more information about an amzn2 image.");
String amiValue = describeImage(ec2, instanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Get a list of instance types.");
String instanceType = getInstanceTypes(ec2);
System.out.println("The instance type is " + instanceType);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("8. Create an instance.");
String newInstanceId = runInstance(ec2, instanceType, keyName, groupName,
amiValue);
System.out.println("The instance Id is " + newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Display information about the running instance. ");
String ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Stop the instance and use a waiter.");
stopInstance(ec2, newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Start the instance and use a waiter.");
startInstance(ec2, newInstanceId);
ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Allocate an Elastic IP address and associate it with
the instance.");
String allocationId = allocateAddress(ec2);
System.out.println("The allocation Id value is " + allocationId);
String associationId = associateAddress(ec2, newInstanceId, allocationId);
System.out.println("The associate Id value is " + associationId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Describe the instance again.");
ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
```



```
System.out.println("14. Disassociate and release the Elastic IP address.");
disassociateAddress(ec2, associationId);
releaseEC2Address(ec2, allocationId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("15. Terminate the instance and use a waiter.");
terminateEC2(ec2, newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("16. Delete the security group.");
deleteEC2SecGroup(ec2, groupId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("17. Delete the key.");
deleteKeys(ec2, keyName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("You successfully completed the Amazon EC2 scenario.");
System.out.println(DASHES);
ec2.close();
}

public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {
    try {
        DeleteSecurityGroupRequest request =
DeleteSecurityGroupRequest.builder()
        .groupId(groupId)
        .build();

        ec2.deleteSecurityGroup(request);
        System.out.println("Successfully deleted security group with Id " +
groupId);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void terminateEC2(Ec2Client ec2, String instanceId) {
```

```
    try {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
            .overrideConfiguration(b -> b.maxAttempts(100))
            .client(ec2)
            .build();

        TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to
terminate. This will take a few minutes.");
        ec2.terminateInstances(ti);
        DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse = ec2Waiter
            .waitUntilInstanceTerminated(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully started instance " + instanceId);
        System.out.println(instanceId + " is terminated!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteKeys(Ec2Client ec2, String keyPair) {
    try {
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        ec2.deleteKeyPair(request);
        System.out.println("Successfully deleted key pair named " + keyPair);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
public static void releaseEC2Address(Ec2Client ec2, String allocId) {
    try {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        ec2.releaseAddress(request);
        System.out.println("Successfully released Elastic IP address " +
allocId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void disassociateAddress(Ec2Client ec2, String associationId) {
    try {
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();

        ec2.disassociateAddress(addressRequest);
        System.out.println("You successfully disassociated the address!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String associateAddress(Ec2Client ec2, String instanceId, String
allocationId) {
    try {
        AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
            .instanceId(instanceId)
            .allocationId(allocationId)
            .build();

        AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
        return associateResponse.associationId();
    }
}
```

```
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String allocateAddress(Ec2Client ec2) {
    try {
        AllocateAddressRequest allocateRequest =
AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        AllocateAddressResponse allocateResponse =
ec2.allocateAddress(allocateRequest);
        return allocateResponse.allocationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void startInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();

    StartInstancesRequest request = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to run. This
will take a few minutes.");
    ec2.startInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();
```

```
    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitForInstanceRunning(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully started instance " + instanceId);
}

public static void stopInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();
    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
will take a few minutes.");
    ec2.stopInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitForInstanceStopped(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully stopped instance " + instanceId);
}

public static String describeEC2Instances(Ec2Client ec2, String newInstanceId) {
    try {
        String pubAddress = "";
        boolean isRunning = false;
        DescribeInstancesRequest request = DescribeInstancesRequest.builder()
            .instanceIds(newInstanceId)
            .build();

        while (!isRunning) {
            DescribeInstancesResponse response = ec2.describeInstances(request);
            String state =
response.reservations().get(0).instances().get(0).state().name().name();
            if (state.compareTo("RUNNING") == 0) {
                System.out.println("Image id is " +
response.reservations().get(0).instances().get(0).imageId());
            }
        }
    }
}
```

```

        System.out.println(
            "Instance type is " +
response.reservations().get(0).instances().get(0).instanceType());
        System.out.println(
            "Instance state is " +
response.reservations().get(0).instances().get(0).state().name());
        pubAddress =
response.reservations().get(0).instances().get(0).publicIpAddress();
        System.out.println("Instance address is " + pubAddress);
        isRunning = true;
    }
}
return pubAddress;
} catch (SsmException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}

public static String runInstance(Ec2Client ec2, String instanceType, String
keyName, String groupName,
    String amiId) {
    try {
        RunInstancesRequest runRequest = RunInstancesRequest.builder()
            .instanceType(instanceType)
            .keyName(keyName)
            .securityGroups(groupName)
            .maxCount(1)
            .minCount(1)
            .imageId(amiId)
            .build();

        System.out.println("Going to start an EC2 instance using a waiter");
        RunInstancesResponse response = ec2.runInstances(runRequest);
        String instanceIdVal = response.instances().get(0).instanceId();
        ec2.waiter().waitUntilInstanceRunning(r ->
r.instanceIds(instanceIdVal));
        System.out.println("Successfully started EC2 instance " + instanceIdVal
+ " based on AMI " + amiId);
        return instanceIdVal;

    } catch (SsmException e) {
        System.err.println(e.getMessage());
    }
}

```

```
        System.exit(1);
    }
    return "";
}

// Get a list of instance types.
public static String getInstanceTypes(Ec2Client ec2) {
    String instanceType;
    try {
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
        .maxResults(10)
        .build();

        DescribeInstanceTypesResponse response =
ec2.describeInstanceTypes(typesRequest);
        List<InstanceTypeInfo> instanceTypes = response.getInstanceTypes();
        for (InstanceTypeInfo type : instanceTypes) {
            System.out.println("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
            System.out.println("Network information is " +
type.networkInfo().toString());
            System.out.println("Instance type is " +
type.getInstanceType().toString());
            instanceType = type.getInstanceType().toString();
            if (instanceType.compareTo("t2.2xlarge") == 0){
                return instanceType;
            }
        }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Display the Description field that corresponds to the instance Id value.
public static String describeImage(Ec2Client ec2, String instanceId) {
    try {
        DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
        .imageIds(instanceId)
        .build();
```

```
        DescribeImagesResponse response = ec2.describeImages(imagesRequest);
        System.out.println("The description of the first image is " +
response.images().get(0).description());
        System.out.println("The name of the first image is " +
response.images().get(0).name());

        // Return the image Id value.
        return response.images().get(0).imageId();

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Get the Id value of an instance with amzn2 in the name.
public static String getParaValues(SsmClient ssmClient) {
    try {
        GetParametersByPathRequest parameterRequest =
GetParametersByPathRequest.builder()
            .path("/aws/service/ami-amazon-linux-latest")
            .build();

        GetParametersByPathIterable responses =
ssmClient.getParametersByPathPaginator(parameterRequest);
        for
(ssoftware.amazon.awssdk.services.ssm.model.GetParametersByPathResponse response :
responses) {
            System.out.println("Test " + response.nextToken());
            List<Parameter> parameterList = response.parameters();
            for (Parameter para : parameterList) {
                System.out.println("The name of the para is: " + para.name());
                System.out.println("The type of the para is: " + para.type());
                if (filterName(para.name())) {
                    return para.value();
                }
            }
        }
    }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```



```
        return "";
    }

    // Return true if the name has amzn2 in it. For example:
    // /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-arm64-gp2
    private static boolean filterName(String name) {
        String[] parts = name.split("/");
        String myValue = parts[4];
        return myValue.contains("amzn2");
    }

    public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
        try {
            DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
                .groupIds(groupId)
                .build();

            // Use a paginator.
            DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
            listGroups.stream()
                .flatMap(r -> r.securityGroups().stream())
                .forEach(group -> System.out
                    .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
        String myIpAddress) {
        try {
            CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
                .groupName(groupName)
                .description(groupDesc)
                .vpcId(vpcId)
                .build();
```

```
        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
        IpRange ipRange = IpRange.builder()
            .cidrIp(myIpAddress + "/0")
            .build();

        IpPermission ipPerm = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(80)
            .fromPort(80)
            .ipRanges(ipRange)
            .build();

        IpPermission ipPerm2 = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(22)
            .fromPort(22)
            .ipRanges(ipRange)
            .build();

        AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(groupName)
            .ipPermissions(ipPerm, ipPerm2)
            .build();

        ec2.authorizeSecurityGroupIngress(authRequest);
        System.out.println("Successfully added ingress policy to security group
" + groupName);
        return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void describeKeys(Ec2Client ec2) {
    try {
        DescribeKeyPairsResponse response = ec2.describeKeyPairs();
        response.keyPairs().forEach(keyPair -> System.out.printf(
            "Found key pair with name %s " +
            "and fingerprint %s",
```

```
        keyPair.keyName(),
        keyPair.keyFingerprint()));

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createKeyPair(Ec2Client ec2, String keyName, String fileName)
{
    try {
        CreateKeyPairRequest request = CreateKeyPairRequest.builder()
            .keyName(keyName)
            .build();

        CreateKeyPairResponse response = ec2.createKeyPair(request);
        String content = response.keyMaterial();
        BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
        writer.write(content);
        writer.close();
        System.out.println("Successfully created key pair named " + keyName);

    } catch (Ec2Exception | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [AllocateAddress](#)
  - [AssociateAddress](#)
  - [AuthorizeSecurityGroupIngress](#)
  - [CreateKeyPair](#)
  - [CreateSecurityGroup](#)
  - [DeleteKeyPair](#)
  - [DeleteSecurityGroup](#)
  - [DescribeImages](#)

- [DescribeInstanceTypes](#)
- [DescribeInstances](#)
- [DescribeKeyPairs](#)
- [DescribeSecurityGroups](#)
- [DisassociateAddress](#)
- [ReleaseAddress](#)
- [RunInstances](#)
- [StartInstances](#)
- [StopInstances](#)
- [TerminateInstances](#)
- [UnmonitorInstances](#)

## Amazon-ECS-Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon ECS verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)

### Aktionen

#### Erstellen eines -Clusters

Das folgende Codebeispiel zeigt, wie Sie einen Amazon-ECS-Cluster erstellen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.ExecuteCommandConfiguration;
import software.amazon.awssdk.services.ecs.model.ExecuteCommandLogging;
import software.amazon.awssdk.services.ecs.model.ClusterConfiguration;
import software.amazon.awssdk.services.ecs.model.CreateClusterResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.CreateClusterRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCluster {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <clusterName>\s

                Where:
                clusterName - The name of the ECS cluster to create.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
```

```
Region region = Region.US_EAST_1;
EcsClient ecsClient = EcsClient.builder()
    .region(region)
    .build();

String clusterArn = createGivenCluster(ecsClient, clusterName);
System.out.println("The cluster ARN is " + clusterArn);
ecsClient.close();
}

public static String createGivenCluster(EcsClient ecsClient, String clusterName)
{
    try {
        ExecuteCommandConfiguration commandConfiguration =
ExecuteCommandConfiguration.builder()
            .logging(ExecuteCommandLogging.DEFAULT)
            .build();

        ClusterConfiguration clusterConfiguration =
ClusterConfiguration.builder()
            .executeCommandConfiguration(commandConfiguration)
            .build();

        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterName(clusterName)
            .configuration(clusterConfiguration)
            .build();

        CreateClusterResponse response =
ecsClient.createCluster(clusterRequest);
        return response.cluster().clusterArn();

    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Weitere API-Informationen finden Sie unter [CreateCluster](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Einen Service erstellen

Das folgende Codebeispiel zeigt, wie Sie einen Amazon-ECS-Service erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.AwsVpcConfiguration;
import software.amazon.awssdk.services.ecs.model.NetworkConfiguration;
import software.amazon.awssdk.services.ecs.model.CreateServiceRequest;
import software.amazon.awssdk.services.ecs.model.LaunchType;
import software.amazon.awssdk.services.ecs.model.CreateServiceResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateService {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <clusterName> <serviceName> <securityGroups>
                <subnets> <taskDefinition>

                Where:
                clusterName - The name of the ECS cluster.
                serviceName - The name of the ECS service to
                create.

                securityGroups - The name of the security group.
                subnets - The name of the subnet.
```

```
        taskDefinition - The name of the task definition.
        """);

    if (args.length != 5) {
        System.out.println(usage);
        System.exit(1);
    }

    String clusterName = args[0];
    String serviceName = args[1];
    String securityGroups = args[2];
    String subnets = args[3];
    String taskDefinition = args[4];
    Region region = Region.US_EAST_1;
    EcsClient ecsClient = EcsClient.builder()
        .region(region)
        .build();

    String serviceArn = createNewService(ecsClient, clusterName,
serviceName, securityGroups, subnets,
        taskDefinition);
    System.out.println("The ARN of the service is " + serviceArn);
    ecsClient.close();
}

public static String createNewService(EcsClient ecsClient,
    String clusterName,
    String serviceName,
    String securityGroups,
    String subnets,
    String taskDefinition) {

    try {
        AwsVpcConfiguration vpcConfiguration =
AwsVpcConfiguration.builder()
            .securityGroups(securityGroups)
            .subnets(subnets)
            .build();

        NetworkConfiguration configuration =
NetworkConfiguration.builder()
            .awsvpcConfiguration(vpcConfiguration)
            .build();
```



```

        CreateServiceRequest serviceRequest =
CreateServiceRequest.builder()
                        .cluster(clusterName)
                        .networkConfiguration(configuration)
                        .desiredCount(1)
                        .launchType(LaunchType.FARGATE)
                        .serviceName(serviceName)
                        .taskDefinition(taskDefinition)
                        .build();

        CreateServiceResponse response =
ecsClient.createService(serviceRequest);
        return response.service().serviceArn();

    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}

```

- Weitere API-Informationen finden Sie unter [CreateService](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Einen Service löschen

Das folgende Codebeispiel zeigt, wie Sie einen Amazon-ECS-Service löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DeleteServiceRequest;
import software.amazon.awssdk.services.ecs.model.EcsException;

```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteService {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <clusterName> <serviceArn>\s

            Where:
                clusterName - The name of the ECS cluster.
                serviceArn - The ARN of the ECS service.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        String serviceArn = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        deleteSpecificService(ecsClient, clusterName, serviceArn);
        ecsClient.close();
    }

    public static void deleteSpecificService(EcsClient ecsClient, String
clusterName, String serviceArn) {
        try {
            DeleteServiceRequest serviceRequest = DeleteServiceRequest.builder()
                .cluster(clusterName)
                .service(serviceArn)

```

```

        .build();

        ecsClient.deleteService(serviceRequest);
        System.out.println("The Service was successfully deleted");

    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

- Weitere API-Informationen finden Sie unter [DeleteService](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beschreiben von Clustern

Das folgende Codebeispiel zeigt, wie Sie Ihre Amazon-ECS-Cluster beschreiben.

### SDK für Java 2.x

#### Note

Auf [GitHub](#) gibt es mehr. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DescribeClustersRequest;
import software.amazon.awssdk.services.ecs.model.DescribeClustersResponse;
import software.amazon.awssdk.services.ecs.model.Cluster;
import software.amazon.awssdk.services.ecs.model.EcsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */

```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DescribeClusters {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <clusterArn> \s

            Where:
            clusterArn - The ARN of the ECS cluster to describe.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterArn = args[0];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        descCluster(ecsClient, clusterArn);
    }

    public static void descCluster(EcsClient ecsClient, String clusterArn) {
        try {
            DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
                .clusters(clusterArn)
                .build();

            DescribeClustersResponse response =
ecsClient.describeClusters(clustersRequest);
            List<Cluster> clusters = response.clusters();
            for (Cluster cluster : clusters) {
                System.out.println("The cluster name is " + cluster.clusterName());
            }
        } catch (EcsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```

    }
  }
}

```

- Weitere API-Informationen finden Sie unter [DescribeClusters](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beschreiben von Aufgaben

Das folgende Codebeispiel zeigt, wie Sie Ihre Amazon-ECS-Aufgaben beschreiben.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DescribeTasksRequest;
import software.amazon.awssdk.services.ecs.model.DescribeTasksResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.Task;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTaskDefinitions {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <clusterArn> <taskId>\s

```

```
        Where:
        clusterArn - The ARN of an ECS cluster.
        taskId - The task Id value.
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String clusterArn = args[0];
    String taskId = args[1];
    Region region = Region.US_EAST_1;
    EcsClient ecsClient = EcsClient.builder()
        .region(region)
        .build();

    getAllTasks(ecsClient, clusterArn, taskId);
    ecsClient.close();
}

public static void getAllTasks(EcsClient ecsClient, String clusterArn, String
taskId) {
    try {
        DescribeTasksRequest tasksRequest = DescribeTasksRequest.builder()
            .cluster(clusterArn)
            .tasks(taskId)
            .build();

        DescribeTasksResponse response = ecsClient.describeTasks(tasksRequest);
        List<Task> tasks = response.tasks();
        for (Task task : tasks) {
            System.out.println("The task ARN is " + task.taskDefinitionArn());
        }

    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [DescribeTasks](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Clustern

Das folgende Codebeispiel zeigt, wie Sie Ihre Amazon-ECS-Cluster auflisten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.ListClustersResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        listAllClusters(ecsClient);
        ecsClient.close();
    }

    public static void listAllClusters(EcsClient ecsClient) {
```

```
    try {
        ListClustersResponse response = ecsClient.listClusters();
        List<String> clusters = response.clusterArns();
        for (String cluster : clusters) {
            System.out.println("The cluster arn is " + cluster);
        }

    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [ListClusters](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Einen Service aktualisieren

Das folgende Codebeispiel zeigt, wie Sie einen Amazon-ECS-Service aktualisieren.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.UpdateServiceRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```



```
*/

public class UpdateService {

    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <clusterName> <serviceArn>\s

            Where:
                clusterName - The cluster name.
                serviceArn - The service ARN value.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        String serviceArn = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        updateSpecificService(ecsClient, clusterName, serviceArn);
        ecsClient.close();
    }

    public static void updateSpecificService(EcsClient ecsClient, String
clusterName, String serviceArn) {
        try {
            UpdateServiceRequest serviceRequest = UpdateServiceRequest.builder()
                .cluster(clusterName)
                .service(serviceArn)
                .desiredCount(0)
                .build();

            ecsClient.updateService(serviceRequest);
            System.out.println("The service was modified");
        }
    }
}
```

```
        } catch (EcsException e) {  
            System.err.println(e.awsErrorDetails().errorMessage());  
            System.exit(1);  
        }  
    }  
}
```

- Weitere API-Informationen finden Sie unter [UpdateService](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beispiele für Elastic Load Balancing mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Elastic Load Balancing verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Erste Schritte

#### Hallo Elastic Load Balancing

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Elastic Load Balancing.

#### SDK für Java 2.x

##### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public class HelloLoadBalancer {
```

```
public static void main(String[] args) {
    ElasticLoadBalancingV2Client loadBalancingV2Client =
ElasticLoadBalancingV2Client.builder()
        .region(Region.US_EAST_1)
        .build();

    DescribeLoadBalancersResponse loadBalancersResponse =
loadBalancingV2Client
        .describeLoadBalancers(r -> r.pageSize(10));
    List<LoadBalancer> loadBalancerList =
loadBalancersResponse.loadBalancers();
    for (LoadBalancer lb : loadBalancerList)
        System.out.println("Load Balancer DNS name = " +
lb.dnsName());
}
```

- Weitere API-Informationen finden Sie unter [DescribeLoadBalancers](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Themen

- [Aktionen](#)
- [Szenarien](#)

## Aktionen

### Erstellen eines Listeners für einen Load Balancer

Das folgende Codebeispiel zeigt, wie Sie einen Listener erstellen, der Anforderungen von einem ELB Load Balancer an eine Zielgruppe weiterleitet.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/*
 * Creates an Elastic Load Balancing load balancer that uses the specified
 * subnets
 * and forwards requests to the specified target group.
 */
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
    String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
            .scheme("internet-facing")
            .build();

        // Create and wait for the load balancer to become available.
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(lbARN)
            .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);
    }
}
```

```
// Create a listener for the load balance.
Action action = Action.builder()
    .targetGroupArn(targetGroupARN)
    .type("forward")
    .build();

CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
    .defaultActions(action)
    .port(port)
    .protocol(protocol)
    .defaultActions(action)
    .build();

getLoadBalancerClient().createListener(listenerRequest);
System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
    + targetGroupARN);

// Return the load balancer DNS name.
return lbDNSName;

} catch (ElasticLoadBalancingV2Exception e) {
    e.printStackTrace();
}
return "";
}
```

- Weitere API-Informationen finden Sie unter [CreateListener](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen einer Zielgruppe

Das folgende Codebeispiel zeigt, wie Sie eine ELB-Zielgruppe erstellen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/*
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
    String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
    System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
    return targetGroupArn;
}
```

- Weitere API-Informationen finden Sie unter [CreateTargetGroup](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erstellen eines Application Load Balancers

Das folgende Codebeispiel zeigt, wie Sie einen ELB Application Load Balancer erstellen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/*
 * Creates an Elastic Load Balancing load balancer that uses the specified
 * subnets
 * and forwards requests to the specified target group.
 */
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
    String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
            .scheme("internet-facing")
            .build();

        // Create and wait for the load balancer to become available.
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(lbARN)
            .build();
```

```
        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
```



- Weitere API-Informationen finden Sie unter [CreateLoadBalancer](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Löschen eines -Load Balancers

Das folgende Codebeispiel zeigt, wie Sie einen ELB Load Balancer löschen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}
```

- Weitere API-Informationen finden Sie unter [DeleteLoadBalancer](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Löschen einer Zielgruppe

Das folgende Codebeispiel zeigt, wie Sie eine ELB-Zielgruppe löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}
```

- Weitere API-Informationen finden Sie unter [DeleteTargetGroup](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Abrufen des Zustands einer Zielgruppe

Das folgende Codebeispiel zeigt, wie Sie den Zustand von Instances in einer ELB-Zielgruppe abrufen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
// Checks the health of the instances in the target group.
public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
    DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
        .names(targetGroupName)
        .build();

    DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

    DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
        .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
        .build();

    DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
    return healthResponse.targetHealthDescriptions();
}
```

- Weitere API-Informationen finden Sie unter [DescribeTargetHealth](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Szenarien

### Erstellen und Verwalten eines ausfallsicheren Services

Das folgende Codebeispiel zeigt, wie Sie einen Webservice mit Load Balancing erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Antwort des Webserver auf Anfragen und Zustandsprüfungen, indem Sie die AWS Systems Manager Parameter aktualisieren.

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\\
\\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\\
\\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\\
\\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\\
\\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
```

```
public static final String policyName = "doc-example-resilience-pol";
public static final String profileName = "doc-example-resilience-prof";

public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

public static final String targetGroupName = "doc-example-resilience-tg";
public static final String autoScalingGroupName = "doc-example-resilience-
group";
public static final String lbName = "doc-example-resilience-lb";
public static final String protocol = "HTTP";
public static final int port = 80;

public static final String DASHES = new String(new char[80]).replace("\0", "-");

public static void main(String[] args) throws IOException, InterruptedException
{
    Scanner in = new Scanner(System.in);
    Database database = new Database();
    AutoScaler autoScaler = new AutoScaler();
    LoadBalancer loadBalancer = new LoadBalancer();

    System.out.println(DASHES);
    System.out.println("Welcome to the demonstration of How to Build and Manage
a Resilient Service!");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("A - SETUP THE RESOURCES");
    System.out.println("Press Enter when you're ready to start deploying
resources.");
    in.nextLine();
    deploy(loadBalancer);
    System.out.println(DASHES);
    System.out.println(DASHES);
    System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    demo(loadBalancer);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("C - DELETE THE RESOURCES");
    System.out.println(""
```

This concludes the demo of how to build and manage a resilient service.

To keep things tidy and to avoid unwanted charges on your account, we can clean up all AWS resources

that were created for this demo.

```
""");
```

```
System.out.println("\n Do you want to delete the resources (y/n)? ");
```

```
String userInput = in.nextLine().trim().toLowerCase(); // Capture user input
```

```
if (userInput.equals("y")) {
```

```
    // Delete resources here
```

```
    deleteResources(loadBalancer, autoScaler, database);
```

```
    System.out.println("Resources deleted.");
```

```
} else {
```

```
    System.out.println("""
```

```
        Okay, we'll leave the resources intact.
```

```
        Don't forget to delete them when you're done with them or you
```

might incur unexpected charges.

```
""");
```

```
}
```

```
System.out.println(DASHES);
```

```
System.out.println(DASHES);
```

```
System.out.println("The example has completed. ");
```

```
System.out.println("\n Thanks for watching!");
```

```
System.out.println(DASHES);
```

```
}
```

```
// Deletes the AWS resources used in this example.
```

```
private static void deleteResources(LoadBalancer loadBalancer, AutoScaler autoScaler, Database database)
```

```
    throws IOException, InterruptedException {
```

```
    loadBalancer.deleteLoadBalancer(lbName);
```

```
    System.out.println("*** Wait 30 secs for resource to be deleted");
```

```
    TimeUnit.SECONDS.sleep(30);
```

```
    loadBalancer.deleteTargetGroup(targetGroupName);
```

```
    autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
```

```
    autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
```

```
    autoScaler.deleteTemplate(templateName);
```

```
    database.deleteTable(tableName);
```

```
}
```

```
private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
    Scanner in = new Scanner(System.in);
    System.out.println(
        """
        For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources
        to set up a load-balanced web service endpoint and explore
some ways to make it resilient
        against various kinds of failures.

        Some of the resources create by this demo are:
        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
        \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.
        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
        """);

    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Creating and populating a DynamoDB table named " +
tableName);
    Database database = new Database();
    database.createTable(tableName, fileName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("""
        Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
        This script starts a Python web server defined in the `server.py`
script. The web server
        listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
        For demo purposes, this server is run as the root user. In
production, the best practice is to
        run a web server, such as Apache, with least-privileged credentials.
    """);
}
```

```
        The template also defines an IAM policy that each instance uses to
assume a role that grants
        permissions to access the DynamoDB recommendation table and Systems
Manager parameters
        that control the flow of the demo.
        """);
```

```
        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println(
            "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);
```

```
        System.out.println("""
            At this point, you have EC2 instances created. Once each instance
starts, it listens for
            HTTP requests. You can see these instances in the console or
continue with the demo.
            Press Enter when you're ready to continue.
            """);
```

```
        in.nextLine();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("""
            Creating an Elastic Load Balancing target group and load balancer.
The target group
```



```

        defines how the load balancer connects to instances. The load
balancer provides a
        single endpoint where clients connect and dispatches requests to
instances in the group.
        """);

    String vpcId = autoScaler.getDefaultVPC();
    List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
    System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
    String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
    String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
    autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
    System.out.println("Verifying access to the load balancer endpoint...");
    boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
    if (!wasSuccessful) {
        System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to "http://checkip.amazonaws.com"
        HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
        try {
            // Execute the request and get the response
            HttpResponse response = httpClient.execute(httpGet);

            // Read the response content.
            String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

            // Print the public IP address.
            System.out.println("Public IP Address: " + ipAddress);
            GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
            if (!groupInfo.isPortOpen()) {
                System.out.println("""
                    For this example to work, the default security group for
your default VPC must
                    allow access from this computer. You can either add it
automatically from this

```

```

        example or add it yourself using the AWS Management
Console.
        """);

        System.out.println(
            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
        System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
        String ans = in.nextLine();
        if ("y".equalsIgnoreCase(ans)) {
            autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
            System.out.println("Security group rule added.");
        } else {
            System.out.println("No security group rule added.");
        }
    }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
    } else if (wasSuccessful) {
        System.out.println("Your load balancer is ready. You can access it by
browsing to:");
        System.out.println("\t http://" + elbDnsName);
    } else {
        System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
        System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
        System.out.println("you can successfully make a GET request to the load
balancer.");
    }

    System.out.println("Press Enter when you're ready to continue with the
demo.");
    in.nextLine();
}

// A method that controls the demo part of the Java program.
public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    ParameterHelper paramHelper = new ParameterHelper();

```

```
System.out.println("Read the ssm_only_policy.json file");
String ssmOnlyPolicy = readFileAsString(ssmJSON);

System.out.println("Resetting parameters to starting values for demo.");
paramHelper.reset();

System.out.println(
    """
        This part of the demonstration shows how to toggle
different parts of the system
        to create situations where the web service fails, and shows
how using a resilient
        architecture can keep the web service running in spite of
these failures.

        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
        """);
demoChoices(loadBalancer);

System.out.println(
    """
        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
        The table name is contained in a Systems Manager parameter
named self.param_helper.table.
        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
        """);
paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

System.out.println(
    """
        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
        """);
demoChoices(loadBalancer);

System.out.println(
    """
        Instead of failing when the recommendation service fails,
the web service can return a static response.
```

```
        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
            Now, sending a GET request to the load balancer endpoint returns a
static response.
            The service still reports as healthy because health checks are still
shallow.
            """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
            Let's also substitute bad credentials for one of the instances in
the target group so that it can't
            access the DynamoDB recommendation table. We will get an instance id
value.
            """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " + profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
            + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
            """"
                Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
```

```

        depending on which instance is selected by the load
balancer.
        """);

demoChoices(loadBalancer);

System.out.println("""
    Let's implement a deep health check. For this demo, a deep health
check tests whether
    the web service can access the DynamoDB table that it depends on for
recommendations. Note that
    the deep health check is only for ELB routing and not for Auto
Scaling instance health.
    This kind of deep health check is not recommended for Auto Scaling
instance health, because it
    risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
    """);

System.out.println("""
    By implementing deep health checks, the load balancer can detect
when one of the instances is failing
    and take that instance out of rotation.
    """);

paramHelper.put(paramHelper.healthCheck, "deep");

System.out.println("""
    Now, checking target health indicates that the instance with bad
credentials
    is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
    instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
    the load balancer takes unhealthy instances out of its rotation.
    """);

demoChoices(loadBalancer);

System.out.println(
    ""
    Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
```

```

        instance is to terminate it and let the auto scaler start a
new instance to replace it.
        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
            Even while the instance is terminating and the new instance is
starting, sending a GET
            request to the web service continues to get a successful
recommendation response because
            the load balancer routes requests to the healthy instances. After
the replacement instance
            starts and reports as healthy, it is included in the load balancing
rotation.

            Note that terminating and replacing an instance typically takes
several minutes, during which time you
            can see the changing health check status until the new instance is
running and healthy.
        """);

        demoChoices(loadBalancer);
        System.out.println(
            "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {

```

```
        System.out.println(i + ": " + actions[i]);
    }

    try {
        System.out.print("\nWhich action would you like to take? ");
        int choice = scanner.nextInt();
        System.out.println("-".repeat(88));

        switch (choice) {
            case 0 -> {
                System.out.println("Request:\n");
                System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                CloseableHttpClient httpClient =
HttpClientBuilder.createDefault();

                // Create an HTTP GET request to the ELB.
                HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);

                // Display the JSON response
                BufferedReader reader = new BufferedReader(
                    new
InputStreamReader(response.getEntity().getContent()));
                StringBuilder jsonResponse = new StringBuilder();
                String line;
                while ((line = reader.readLine()) != null) {
                    jsonResponse.append(line);
                }
                reader.close();

                // Print the formatted JSON response.
                System.out.println("Full Response:\n");
                System.out.println(jsonResponse.toString());

                // Close the HTTP client.
                httpClient.close();
            }
        }
    }
```

```

        case 1 -> {
            System.out.println("\nChecking the health of load balancer
targets:\n");
            List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
            for (TargetHealthDescription target : health) {
                System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                                target.target().port(),
target.targetHealth().stateAsString());
            }
            System.out.println("""
Note that it can take a minute or two for the health
check to update
                                after changes are made.
                                """);
        }
        case 2 -> {
            System.out.println("\nOkay, let's move on.");
            System.out.println("-".repeat(88));
            return; // Exit the method when choice is 2
        }
        default -> System.out.println("You must choose a value between
0-2. Please select again.");
    }

    } catch (java.util.InputMismatchException e) {
        System.out.println("Invalid input. Please select again.");
        scanner.nextLine(); // Clear the input buffer.
    }
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}

```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
public class AutoScaler {
```



```
private static Ec2Client ec2Client;
private static AutoScalingClient autoScalingClient;
private static IamClient iamClient;

private static SsmClient ssmClient;

private IamClient getIAMClient() {
    if (iamClient == null) {
        iamClient = IamClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return iamClient;
}

private SsmClient getSSMClient() {
    if (ssmClient == null) {
        ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ssmClient;
}

private Ec2Client getEc2Client() {
    if (ec2Client == null) {
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
```

```
    * Terminates and instances in an EC2 Auto Scaling group. After an instance is
    * terminated, it can no longer be accessed.
    */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
        TerminateInstanceInAutoScalingGroupRequest
            .builder()
            .instanceId(instanceId)
            .shouldDecrementDesiredCapacity(false)
            .build();

        getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
    newInstanceProfileName, String profileAssociationId)
        throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
        iamInstanceProfile =
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
            .builder()
            .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
            // name.
            .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
        ReplaceIamInstanceProfileAssociationRequest
            .builder()
            .iamInstanceProfile(iamInstanceProfile)
            .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
            .build();
    }
}
```

```
try {
    getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
    // Handle the response as needed.
} catch (Ec2Exception e) {
    // Handle exceptions, log, or report the error.
    System.err.println("Error: " + e.getMessage());
}
System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
    newInstanceProfileName);
TimeUnit.SECONDS.sleep(15);
boolean instReady = false;
int tries = 0;

// Reboot after 60 seconds
while (!instReady) {
    if (tries % 6 == 0) {
        getEc2Client().rebootInstances(RebootInstancesRequest.builder()
            .instanceIds(instanceId)
            .build());
        System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
    }
    tries++;
    try {
        TimeUnit.SECONDS.sleep(10);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
    List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
    for (InstanceInformation info : instanceInformationList) {
        if (info.instanceId().equals(instanceId)) {
            instReady = true;
            break;
        }
    }
}

SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
```

```

        .instanceIds(instanceId)
        .documentName("AWS-RunShellScript")
        .parameters(Collections.singletonMap("commands",
            Collections.singletonList("cd / && sudo python3 server.py
80"))))
        .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(secGroupId)
            .cidrIp(ipAddress)
            .fromPort(Integer.parseInt(port))
            .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
            .builder()
            .instanceProfileName(profileName)
            .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.getInstanceProfile().getInstanceProfileName();
            System.out.println(name);
        }
    }

```

```
        RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
    .instanceProfileName(profileName)
    .roleName(roleName)
    .build();

        getIAMClient().removeRoleFromInstanceProfile(profileRequest);
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
    .instanceProfileName(profileName)
    .build();

        getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
        System.out.println("Deleted instance profile " + profileName);

        DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
    .roleName(roleName)
    .build();

        // List attached role policies.
        ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
    .listAttachedRolePolicies(role -> role.roleName(roleName));
        List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
        for (AttachedPolicy attachedPolicy : attachedPolicies) {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
    .roleName(roleName)
    .policyArn(attachedPolicy.policyArn())
    .build();

            getIAMClient().detachRolePolicy(request);
            System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
        }

        getIAMClient().deleteRole(deleteRoleRequest);
        System.out.println("Instance profile and role deleted.");

    } catch (IamException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
    System.out.format(templateName + " was deleted.");
}

public void deleteAutoScaleGroup(String groupName) {
    DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
        .autoScalingGroupName(groupName)
        .forceDelete(true)
        .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println(groupName + " was deleted.");
}

/*
 * Verify the default security group of the specified VPC allows ingress from
 * this
 * computer. This can be done by allowing ingress from this computer's IP
 * address. In some situations, such as connecting from a corporate network, you
 * must instead specify a prefix list ID. You can also temporarily open the port
 * to
 * any IP address while running this example. If you do, be sure to remove
 * public
 * access when you're done.
 *
 */
public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
    boolean portIsOpen = false;
    GroupInfo groupInfo = new GroupInfo();
    try {
        Filter filter = Filter.builder()
            .name("group-name")
            .values("default")
            .build();

        Filter filter1 = Filter.builder()
            .name("vpc-id")
            .values(VPC)
            .build();
```

```

DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
    .filters(filter, filter1)
    .build();

DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
    .describeSecurityGroups(securityGroupsRequest);

String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
groupInfo.setGroupName(securityGroup);

for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
    System.out.println("Found security group: " + secGroup.groupId());

    for (IpPermission ipPermission : secGroup.ipPermissions()) {
        if (ipPermission.fromPort() == port) {
            System.out.println("Found inbound rule: " + ipPermission);
            for (IpRange ipRange : ipPermission.ipRanges()) {
                String cidrIp = ipRange.cidrIp();
                if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                    System.out.println(cidrIp + " is applicable");
                    portIsOpen = true;
                }
            }

            if (!ipPermission.prefixListIds().isEmpty()) {
                System.out.println("Prefix lList is applicable");
                portIsOpen = true;
            }

            if (!portIsOpen) {
                System.out
                    .println("The inbound rule does not appear to be
open to either this computer's IP,"
                    + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
            } else {
                break;
            }
        }
    }
}
}

```

```
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }

    groupInfo.setPortOpen(portIsOpen);
    return groupInfo;
}

/**
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {
        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
            .autoScalingGroupName(asGroupName)
            .targetGroupARNs(targetGroupARN)
            .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
        System.out.println("Attached load balancer to " + asGroupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

    // Get availability zones.
    software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
            .builder()
            .build();
```



```
DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
    .collect(Collectors.toList());

String availabilityZones = String.join(",", availabilityZoneNames);
LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
    .launchTemplateName(templateName)
    .version("$Default")
    .build();

String[] zones = availabilityZones.split(",");
CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
    .launchTemplate(specification)
    .availabilityZones(zones)
    .maxSize(groupSize)
    .minSize(groupSize)
    .autoScalingGroupName(autoScalingGroupName)
    .build();

try {
    getAutoScalingClient().createAutoScalingGroup(groupRequest);
} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
return zones;
}

public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();
```

```
        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
        .builder()
        .filters(defaultFilter)
        .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
            .name("vpc-id")
            .values(vpcId)
            .build();

        Filter azFilter = Filter.builder()
            .name("availability-zone")
            .values(availabilityZones)
            .build();

        Filter defaultForAZ = Filter.builder()
            .name("default-for-az")
            .values("true")
            .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
            .filters(vpcFilter, azFilter, defaultForAZ)
            .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();
```

```
DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
List<String> instanceIds = autoScalingGroup.instances().stream()
    .map(instance -> instance.instanceId())
    .collect(Collectors.toList());

String[] instanceIdArray = instanceIds.toArray(new String[0]);
for (String instanceId : instanceIdArray) {
    System.out.println("Instance ID: " + instanceId);
    return instanceId;
}
return "";
}

// Gets data about the profile associated with an instance.
public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();

    DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
        .describeIamInstanceProfileAssociations(associationsRequest);
    return response.iamInstanceProfileAssociations().get(0).associationId();
}

public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
    ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
    ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
    for (Policy policy : listPoliciesResponse.policies()) {
        if (policy.policyName().equals(policyName)) {
            // List the entities (users, groups, roles) that are attached to the
policy.

```

```
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
    .builder()
    .policyArn(policy.arn())
    .build();
    ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
        .listEntitiesForPolicy(listEntitiesRequest);
    if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
        || !listEntitiesResponse.policyRoles().isEmpty()) {
        // Detach the policy from any entities it is attached to.
        DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
            .policyArn(policy.arn())
            .roleName(roleName) // Specify the name of the IAM role
            .build();

        getIAMClient().detachRolePolicy(detachPolicyRequest);
        System.out.println("Policy detached from entities.");
    }

    // Now, you can delete the policy.
    DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
    .policyArn(policy.arn())
    .build();

    getIAMClient().deletePolicy(deletePolicyRequest);
    System.out.println("Policy deleted successfully.");
    break;
}
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
```

```

        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .roleName(roleName) // Remove the extra dot here
                .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
            .instanceProfileName(InstanceProfile)
            .build();

        getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}

```

Erstellen Sie eine Klasse, die Elastic-Load-Balancing-Aktionen beinhaltet.

```

public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.

```

```
public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
    DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
        .names(targetGroupName)
        .build();

    DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

    DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
        .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
        .build();

    DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
    return healthResponse.targetHealthDescriptions();
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName) {
    DescribeLoadBalancersResponse res = getLoadBalancerClient()
        .describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
```

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

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to the ELB.
    HttpGet httpGet = new HttpGet("http://" + elbDnsName);
    try {
        while ((!success) && (retries > 0)) {
            // Execute the request and get the response.
            HttpResponse response = httpClient.execute(httpGet);
            int statusCode = response.getStatusLine().getStatusCode();
            System.out.println("HTTP Status Code: " + statusCode);
            if (statusCode == 200) {
                success = true;
            } else {
                retries--;
            }
        }
    }
}
```

```
        System.out.println("Got connection error from load balancer
endpoint, retrying...");
        TimeUnit.SECONDS.sleep(15);
    }
}

} catch (org.apache.http.conn.HttpHostConnectException e) {
    System.out.println(e.getMessage());
}

System.out.println("Status.." + success);
return success;
}

/*
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
    String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
    System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
    return targetGroupArn;
}

/*
```



```
* Creates an Elastic Load Balancing load balancer that uses the specified
* subnets
* and forwards requests to the specified target group.
*/
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
    String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
            .scheme("internet-facing")
            .build();

        // Create and wait for the load balancer to become available.
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(lbARN)
            .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
```

```

        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}

```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```

public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }
}

```

```
}

// Checks to see if the Amazon DynamoDB table exists.
private boolean doesTableExist(String tableName) {
    try {
        // Describe the table and catch any exceptions.
        DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        getDynamoDbClient().describeTable(describeTableRequest);
        System.out.println("Table '" + tableName + "' exists.");
        return true;

    } catch (ResourceNotFoundException e) {
        System.out.println("Table '" + tableName + "' does not exist.");
    } catch (DynamoDbException e) {
        System.err.println("Error checking table existence: " + e.getMessage());
    }
    return false;
}

/**
 * Creates a DynamoDB table to use a recommendation service. The table has a
 * hash key named 'MediaType' that defines the type of media recommended, such
 * as
 * Book or Movie, and a range key named 'ItemId' that, combined with the
 * MediaType,
 * forms a unique identifier for the recommended item.
 */
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()

```

```

        .attributeName("ItemId")
        .attributeType(ScalarAttributeType.N)
        .build())
    .keySchema(
        KeySchemaElement.builder()
            .attributeName("MediaType")
            .keyType(KeyType.HASH)
            .build(),
        KeySchemaElement.builder()
            .attributeName("ItemId")
            .keyType(KeyType.RANGE)
            .build())
    .provisionedThroughput(
        ProvisionedThroughput.builder()
            .readCapacityUnits(5L)
            .writeCapacityUnits(5L)
            .build())
    .build();

getDynamoDbClient().createTable(createTableRequest);
System.out.println("Creating table " + tableName + "...");

// Wait until the Amazon DynamoDB table is created.
DescribeTableRequest tableRequest = DescribeTableRequest.builder()
    .tableName(tableName)
    .build();

WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("Table " + tableName + " created.");

// Add records to the table.
populateTable(fileName, tableName);
}
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.

```

```

public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
    System.out.println("Added all records to the " + tableName);
}
}

```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```

public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
    }
}

```

```
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
            .overwrite(true)
            .type("String")
            .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [AttachLoadBalancerTargetGroups](#)
  - [CreateAutoScalingGroup](#)
  - [CreateInstanceProfile](#)
  - [CreateLaunchTemplate](#)
  - [CreateListener](#)
  - [CreateLoadBalancer](#)
  - [CreateTargetGroup](#)
  - [DeleteAutoScalingGroup](#)
  - [DeleteInstanceProfile](#)
  - [DeleteLaunchTemplate](#)
  - [DeleteLoadBalancer](#)
  - [DeleteTargetGroup](#)
  - [DescribeAutoScalingGroups](#)
  - [DescribeAvailabilityZones](#)

- [DescribeInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplaceInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

## MediaStore -Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit verwenden MediaStore.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)

### Aktionen

Erstellen eines Containers

Das folgende Codebeispiel zeigt, wie Sie einen - AWS Elemental MediaStore Container erstellen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <containerName>

            Where:
                containerName - The name of the container to create.
            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
```



```

        .region(region)
        .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }

    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
String containerName) {
        try {
            CreateContainerRequest containerRequest =
CreateContainerRequest.builder()
                .containerName(containerName)
                .build();

            CreateContainerResponse containerResponse =
mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
containerResponse.container().arn());
            System.out.println("Finished ");

        } catch (MediaStoreException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}

```

- Weitere API-Informationen finden Sie unter [CreateContainer](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen eines Containers

Das folgende Codebeispiel zeigt, wie Sie einen - AWS Elemental MediaStore Container löschen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <containerName>

            Where:
                containerName - The name of the container to create.
            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
```

```

        .region(region)
        .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }

    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
String containerName) {
        try {
            CreateContainerRequest containerRequest =
CreateContainerRequest.builder()
                .containerName(containerName)
                .build();

            CreateContainerResponse containerResponse =
mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
containerResponse.container().arn());
            System.out.println("Finished ");

        } catch (MediaStoreException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}

```

- Weitere API-Informationen finden Sie unter [DeleteContainer](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Ein Objekt löschen

Das folgende Codebeispiel zeigt, wie Sie ein AWS Elemental MediaStore Objekt löschen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.DeleteObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = ""

                Usage:    <completePath> <containerName>

                Where:
                    completePath - The path (including the container) of the item to
delete.
                    containerName - The name of the container.

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String completePath = args[0];
String containerName = args[1];
Region region = Region.US_EAST_1;
URI uri = new URI(getEndpoint(containerName));

MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
    .endpointOverride(uri)
    .region(region)
    .build();

deleteMediaObject(mediaStoreData, completePath);
mediaStoreData.close();
}

public static void deleteMediaObject(MediaStoreDataClient mediaStoreData, String
completePath) {
    try {
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
            .path(completePath)
            .build();

        mediaStoreData.deleteObject(deleteObjectRequest);

    } catch (MediaStoreDataException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

private static String getEndpoint(String containerName) {
    Region region = Region.US_EAST_1;
    MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
        .region(region)
        .build();

    DescribeContainerRequest containerRequest =
DescribeContainerRequest.builder()
    .containerName(containerName)
    .build();

    DescribeContainerResponse response =
mediaStoreClient.describeContainer(containerRequest);
mediaStoreClient.close();
}
```

```
        return response.container().endpoint();
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteObject](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beschreiben eines Containers

Das folgende Codebeispiel zeigt, wie Sie einen AWS Elemental MediaStore Container beschreiben.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeContainer {

    public static void main(String[] args) {
        final String usage = ""

                Usage:    <containerName>

                Where:
```

```
        containerName - The name of the container to describe.
        """);

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String containerName = args[0];
    Region region = Region.US_EAST_1;
    MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
        .region(region)
        .build();

    System.out.println("Status is " + checkContainer(mediaStoreClient,
containerName));
    mediaStoreClient.close();
}

public static String checkContainer(MediaStoreClient mediaStoreClient, String
containerName) {
    try {
        DescribeContainerRequest describeContainerRequest =
DescribeContainerRequest.builder()
            .containerName(containerName)
            .build();

        DescribeContainerResponse containerResponse =
mediaStoreClient.describeContainer(describeContainerRequest);
        System.out.println("The container name is " +
containerResponse.container().name());
        System.out.println("The container ARN is " +
containerResponse.container().arn());
        return containerResponse.container().status().toString();

    } catch (MediaStoreException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Weitere API-Informationen finden Sie unter [DescribeContainer](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Abrufen eines Objekts

Das folgende Codebeispiel zeigt, wie Sie ein AWS Elemental MediaStore Objekt abrufen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectResponse;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = ""
```



```
Usage:    <completePath> <containerName> <savePath>
```

```
Where:
```

```
    completePath - The path of the object in the container (for  
example, Videos5/sampleVideo.mp4).
```

```
    containerName - The name of the container.
```

```
    savePath - The path on the local drive where the file is saved,  
including the file name (for example, C:/AWS/myvid.mp4).
```

```
""";
```

```
if (args.length != 3) {  
    System.out.println(usage);  
    System.exit(1);  
}
```

```
String completePath = args[0];  
String containerName = args[1];  
String savePath = args[2];
```

```
Region region = Region.US_EAST_1;  
URI uri = new URI(getEndpoint(containerName));  
MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()  
    .endpointOverride(uri)  
    .region(region)  
    .build();
```

```
getMediaObject(mediaStoreData, completePath, savePath);  
mediaStoreData.close();
```

```
}
```

```
public static void getMediaObject(MediaStoreDataClient mediaStoreData, String  
completePath, String savePath) {
```

```
    try {  
        GetObjectRequest objectRequest = GetObjectRequest.builder()  
            .path(completePath)  
            .build();
```

```
        // Write out the data to a file.  
        ResponseInputStream<GetObjectResponse> data =  
mediaStoreData.getObject(objectRequest);  
        byte[] buffer = new byte[data.available()];  
        data.read(buffer);
```

```
        File targetFile = new File(savePath);
        OutputStream outputStream = new FileOutputStream(targetFile);
        outputStream.write(buffer);
        System.out.println("The data was written to " + savePath);

    } catch (MediaStoreDataException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

private static String getEndpoint(String containerName) {
    Region region = Region.US_EAST_1;
    MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
        .region(region)
        .build();

    DescribeContainerRequest containerRequest =
DescribeContainerRequest.builder()
        .containerName(containerName)
        .build();

    DescribeContainerResponse response =
mediaStoreClient.describeContainer(containerRequest);
    return response.container().endpoint();
}
}
```

- Weitere API-Informationen finden Sie unter [GetObject](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Container auflisten

Das folgende Codebeispiel zeigt, wie AWS Elemental MediaStore Container aufgelistet werden.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.Container;
import software.amazon.awssdk.services.mediastore.model.ListContainersResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListContainers {

    public static void main(String[] args) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
            .region(region)
            .build();

        listAllContainers(mediaStoreClient);
        mediaStoreClient.close();
    }

    public static void listAllContainers(MediaStoreClient mediaStoreClient) {
        try {
            ListContainersResponse containersResponse =
mediaStoreClient.listContainers();
            List<Container> containers = containersResponse.containers();

```

```
        for (Container container : containers) {
            System.out.println("Container name is " + container.name());
        }

    } catch (MediaStoreException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [ListContainers](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Einfügen eines Objekts in einen Container

Das folgende Codebeispiel zeigt, wie Sie ein Objekt in einen AWS Elemental MediaStore Container einfügen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectResponse;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import java.io.File;
import java.net.URI;
import java.net.URISyntaxException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutObject {
    public static void main(String[] args) throws URISyntaxException {
        final String USAGE = ""

            To run this example, supply the name of a container, a file location
            to use, and path in the container\s

                Ex: <containerName> <filePath> <completePath>
                """;

        if (args.length < 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String containerName = args[0];
        String filePath = args[1];
        String completePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
            .endpointOverride(uri)
            .region(region)
            .build();

        putMediaObject(mediaStoreData, filePath, completePath);
        mediaStoreData.close();
    }

    public static void putMediaObject(MediaStoreDataClient mediaStoreData, String
filePath, String completePath) {
        try {
            File myFile = new File(filePath);
            RequestBody requestBody = RequestBody.fromFile(myFile);
```

```

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .path(completePath)
            .contentType("video/mp4")
            .build();

        PutObjectResponse response = mediaStoreData.putObject(objectRequest,
requestBody);
        System.out.println("The saved object is " +
response.storageClass().toString());

    } catch (MediaStoreDataException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String getEndpoint(String containerName) {

    Region region = Region.US_EAST_1;
    MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
        .region(region)
        .build();

    DescribeContainerRequest containerRequest =
DescribeContainerRequest.builder()
        .containerName(containerName)
        .build();

    DescribeContainerResponse response =
mediaStoreClient.describeContainer(containerRequest);
    return response.container().endpoint();
}
}

```

- Weitere API-Informationen finden Sie unter [PutObject](#) in der APIAWS SDK for Java 2.x - Referenz für .

## OpenSearch Servicebeispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit OpenSearch Service verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)

## Aktionen

Domain erstellen

Das folgende Codebeispiel zeigt, wie Sie eine - OpenSearch Service-Domain erstellen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.EBSOptions;
import software.amazon.awssdk.services.opensearch.model.VolumeType;
import software.amazon.awssdk.services.opensearch.model.NodeToNodeEncryptionOptions;
import software.amazon.awssdk.services.opensearch.model.CreateDomainRequest;
import software.amazon.awssdk.services.opensearch.model.CreateDomainResponse;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateDomain {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <domainName>

            Where:
                domainName - The name of the domain to create.
            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainName = args[0];
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
            .region(region)
            .build();

        createNewDomain(searchClient, domainName);
        System.out.println("Done");
    }

    public static void createNewDomain(OpenSearchClient searchClient, String
domainName) {
        try {
            ClusterConfig clusterConfig = ClusterConfig.builder()
                .dedicatedMasterEnabled(true)
                .dedicatedMasterCount(3)
                .dedicatedMasterType("t2.small.search")
                .instanceType("t2.small.search")
                .instanceCount(5)
                .build();

            EBSOptions ebsOptions = EBSOptions.builder()
                .ebsEnabled(true)
                .volumeSize(10)
```



```
        .volumeType(VolumeType.GP2)
        .build();

    NodeToNodeEncryptionOptions encryptionOptions =
NodeToNodeEncryptionOptions.builder()
        .enabled(true)
        .build();

    CreateDomainRequest domainRequest = CreateDomainRequest.builder()
        .domainName(domainName)
        .engineVersion("OpenSearch_1.0")
        .clusterConfig(clusterConfig)
        .ebsOptions(ebsOptions)
        .nodeToNodeEncryptionOptions(encryptionOptions)
        .build();

    System.out.println("Sending domain creation request...");
    CreateDomainResponse createResponse =
searchClient.createDomain(domainRequest);
    System.out.println("Domain status is " +
createResponse.domainStatus().toString());
    System.out.println("Domain Id is " +
createResponse.domainStatus().domainId());

    } catch (OpenSearchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [CreateDomain](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Domäne löschen

Das folgende Codebeispiel zeigt, wie Sie eine - OpenSearch Service-Domäne löschen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDomain {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <domainName>

                Where:
                domainName - The name of the domain to delete.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainName = args[0];
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
                .region(region)
                .build();
```

```
        deleteSpecificDomain(searchClient, domainName);
        System.out.println("Done");
    }

    public static void deleteSpecificDomain(OpenSearchClient searchClient, String
domainName) {
        try {
            DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
                .domainName(domainName)
                .build();

            searchClient.deleteDomain(domainRequest);
            System.out.println(domainName + " was successfully deleted.");

        } catch (OpenSearchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteDomain](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Domains

Das folgende Codebeispiel zeigt, wie Sie OpenSearch Service-Domains auflisten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.DomainInfo;
```

```
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesRequest;
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesResponse;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListDomainNames {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();
        listAllDomains(searchClient);
        System.out.println("Done");
    }

    public static void listAllDomains(OpenSearchClient searchClient) {
        try {
            ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()
                .engineType("OpenSearch")
                .build();

            ListDomainNamesResponse response =
searchClient.listDomainNames(namesRequest);
            List<DomainInfo> domainInfoList = response.domainNames();
            for (DomainInfo domain : domainInfoList)
                System.out.println("Domain name is " + domain.domainName());

        } catch (OpenSearchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [ListDomainNames](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Ändern einer Cluster-Konfiguration

Das folgende Codebeispiel zeigt, wie Sie eine Cluster-Konfiguration einer - OpenSearch Service-Domain ändern.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigRequest;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateDomain {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <domainName>

                Where:
```

```
        domainName - The name of the domain to update.

        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String domainName = args[0];
    Region region = Region.US_EAST_1;
    OpenSearchClient searchClient = OpenSearchClient.builder()
        .region(region)
        .build();

    updateSpecificDomain(searchClient, domainName);
    System.out.println("Done");
}

public static void updateSpecificDomain(OpenSearchClient searchClient, String
domainName) {
    try {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .instanceCount(3)
            .build();

        UpdateDomainConfigRequest updateDomainConfigRequest =
UpdateDomainConfigRequest.builder()
            .domainName(domainName)
            .clusterConfig(clusterConfig)
            .build();

        System.out.println("Sending domain update request...");
        UpdateDomainConfigResponse updateResponse =
searchClient.updateDomainConfig(updateDomainConfigRequest);
        System.out.println("Domain update response from Amazon OpenSearch
Service:");
        System.out.println(updateResponse.toString());

    } catch (OpenSearchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- Weitere API-Informationen finden Sie unter [UpdateDomainConfig](#) in der APIAWS SDK for Java 2.x -Referenz für .

## EventBridge -Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit verwenden EventBridge.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Erste Schritte

#### Hallo EventBridge

Die folgenden Codebeispiele veranschaulichen, wie Sie mit der Verwendung von EventBridge beginnen.

#### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.
```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
*/
public class HelloEventBridge {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        EventBridgeClient eventBrClient = EventBridgeClient.builder()
            .region(region)
            .build();

        listBuses(eventBrClient);
        eventBrClient.close();
    }

    public static void listBuses(EventBridgeClient eventBrClient) {
        try {
            ListEventBusesRequest busesRequest = ListEventBusesRequest.builder()
                .limit(10)
                .build();

            ListEventBusesResponse response =
eventBrClient.listEventBuses(busesRequest);
            List<EventBus> buses = response.eventBuses();
            for (EventBus bus : buses) {
                System.out.println("The name of the event bus is: " + bus.name());
                System.out.println("The ARN of the event bus is: " + bus.arn());
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
}
```

- Weitere API-Informationen finden Sie unter [ListEventBuses](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Themen



- [Aktionen](#)
- [Szenarien](#)

## Aktionen

### Hinzufügen eines Ziels

Das folgende Codebeispiel zeigt, wie Sie einem Amazon- EventBridge Ereignis ein Ziel hinzufügen.

SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Fügen Sie ein Amazon-SNS-Thema als Ziel für eine Regel hinzu.

```
// Add a rule which triggers an SNS target when a file is uploaded to an S3
// bucket.
public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn,
    String topicName, String eventRuleName, String bucketName) {
    String targetID = java.util.UUID.randomUUID().toString();
    Target myTarget = Target.builder()
        .id(targetID)
        .arn(topicArn)
        .build();

    List<Target> targets = new ArrayList<>();
    targets.add(myTarget);
    PutTargetsRequest request = PutTargetsRequest.builder()
        .eventBusName(null)
        .targets(targets)
        .rule(ruleName)
        .build();

    eventBrClient.putTargets(request);
    System.out.println("Added event rule " + eventRuleName + " with Amazon SNS
target " + topicName + " for bucket ")
```

```

        + bucketName + ".");
    }

```

Fügen Sie einen Eingabe-Transformator als Ziel für eine Regel hinzu.

```

public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
    String ruleName) {
    String targetId = java.util.UUID.randomUUID().toString();
    InputTransformer inputTransformer = InputTransformer.builder()
        .inputTemplate("\\"Notification: sample event was received.\\"")
        .build();

    Target target = Target.builder()
        .id(targetId)
        .arn(topicArn)
        .inputTransformer(inputTransformer)
        .build();

    try {
        PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
            .rule(ruleName)
            .targets(target)
            .eventBusName(null)
            .build();

        eventBrClient.putTargets(targetsRequest);
    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Weitere API-Informationen finden Sie unter [PutTargets](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen einer Regel

Das folgende Codebeispiel zeigt, wie Sie eine Amazon- EventBridge Regel erstellen.

## SDK für Java 2.x

**Note**

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Erstellen Sie eine geplante Regel.

```
public static void createEBRule(EventBridgeClient eventBrClient, String
ruleName, String cronExpression) {
    try {
        PutRuleRequest ruleRequest = PutRuleRequest.builder()
            .name(ruleName)
            .eventBusName("default")
            .scheduleExpression(cronExpression)
            .state("ENABLED")
            .description("A test rule that runs on a schedule created by the
Java API")
            .build();

        PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
        System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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.
public static void addEventRule(EventBridgeClient eventBrClient, String roleArn,
String bucketName,
    String eventRuleName) {
    String pattern = "{\n" +
```

```

        "  \"source\": [\"aws.s3\"],\n" +
        "  \"detail-type\": [\"Object Created\"],\n" +
        "  \"detail\": {\n" +
        "    \"bucket\": {\n" +
        "      \"name\": [\"\" + bucketName + "\"]\n" +
        "    }\n" +
        "  }\n" +
        "};

try {
    PutRuleRequest ruleRequest = PutRuleRequest.builder()
        .description("Created by using the AWS SDK for Java v2")
        .name(eventRuleName)
        .eventPattern(pattern)
        .roleArn(roleArn)
        .build();

    PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
    System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Weitere API-Informationen finden Sie unter [PutRule](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Löschen einer Regel

Das folgende Codebeispiel zeigt, wie Sie eine Amazon- EventBridge Regel löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
    DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
        .name(ruleName)
        .build();

    eventBrClient.deleteRule(ruleRequest);
    System.out.println("Successfully deleted the rule");
}
```

- Weitere API-Informationen finden Sie unter [DeleteRule](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beschreiben einer Regel

Das folgende Codebeispiel zeigt, wie Sie eine Amazon- EventBridge Regel beschreiben.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    try {
        DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
            .name(eventRuleName)
            .build();

        DescribeRuleResponse response = eventBrClient.describeRule(ruleRequest);
        System.out.println("The state of the rule is " +
response.stateAsString());

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- Weitere API-Informationen finden Sie unter [DescribeRule](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Deaktivieren einer Regel

Das folgende Codebeispiel zeigt, wie Sie eine Amazon- EventBridge Regel deaktivieren.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Deaktivieren Sie eine Regel anhand ihres Regelnamens.

```
public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
    try {
        if (!isEnabled) {
            System.out.println("Disabling the rule: " + eventRuleName);
            DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                .name(eventRuleName)
                .build();

            eventBrClient.disableRule(ruleRequest);
        } else {
            System.out.println("Enabling the rule: " + eventRuleName);
            EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                .name(eventRuleName)
                .build();
            eventBrClient.enableRule(ruleRequest);
        }
    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
}
```

- Weitere API-Informationen finden Sie unter [DisableRule](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Aktivieren einer Regel

Das folgende Codebeispiel zeigt, wie Sie eine Amazon- EventBridge Regel aktivieren.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Aktivieren Sie eine Regel anhand ihres Regelnamens.

```
public static void changeRuleState(EventBridgeClient eventBrClient, String  
eventRuleName, Boolean isEnabled) {  
    try {  
        if (!isEnabled) {  
            System.out.println("Disabling the rule: " + eventRuleName);  
            DisableRuleRequest ruleRequest = DisableRuleRequest.builder()  
                .name(eventRuleName)  
                .build();  
  
            eventBrClient.disableRule(ruleRequest);  
        } else {  
            System.out.println("Enabling the rule: " + eventRuleName);  
            EnableRuleRequest ruleRequest = EnableRuleRequest.builder()  
                .name(eventRuleName)  
                .build();  
            eventBrClient.enableRule(ruleRequest);  
        }  
    }  
  
    } catch (EventBridgeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
    }  
}
```

```

        System.exit(1);
    }
}

```

- Weitere API-Informationen finden Sie unter [EnableRule](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Regelnamen für ein Ziel

Das folgende Codebeispiel zeigt, wie Sie Amazon- EventBridge Regelnamen für ein Ziel auflisten.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Listen Sie alle Regelnamen mithilfe des Ziels auf.

```

public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
    ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
        .targetArn(topicArn)
        .build();

    ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
    List<String> rules = response.ruleNames();
    for (String rule : rules) {
        System.out.println("The rule name is " + rule);
    }
}
}

```

- Weitere API-Informationen finden Sie unter [ListRuleNamesByTarget](#) in der APIAWS SDK for Java 2.x -Referenz für .



## Auflisten von Regeln

Das folgende Codebeispiel zeigt, wie Sie Amazon- EventBridge Regeln auflisten.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Aktivieren Sie eine Regel anhand ihres Regelnamens.

```
public static void listRules(EventBridgeClient eventBrClient) {
    try {
        ListRulesRequest rulesRequest = ListRulesRequest.builder()
            .eventBusName("default")
            .limit(10)
            .build();

        ListRulesResponse response = eventBrClient.listRules(rulesRequest);
        List<Rule> rules = response.rules();
        for (Rule rule : rules) {
            System.out.println("The rule name is : " + rule.name());
            System.out.println("The rule description is : " +
rule.description());
            System.out.println("The rule state is : " + rule.stateAsString());
        }

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [ListRules](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Zielen für eine Regel

Das folgende Codebeispiel zeigt, wie Sie Amazon- EventBridge Ziele für eine Regel auflisten.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Listen Sie alle Ziele für eine Regel mithilfe des Regelnamens auf.

```
public static void listTargets(EventBridgeClient eventBrClient, String ruleName)
{
    ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
        .rule(ruleName)
        .build();


    ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
    List<Target> targetsList = res.targets();
    for (Target target: targetsList) {
        System.out.println("Target ARN: "+target.arn());
    }
}
```

- Weitere API-Informationen finden Sie unter [ListTargetsByRule](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Entfernen von Zielen aus einer Regel

Das folgende Codebeispiel zeigt, wie Sie Amazon- EventBridge Ziele aus einer Regel entfernen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Entfernen Sie alle Ziele für eine Regel mithilfe des Regelnamens.

```
public static void deleteTargetsFromRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    // First, get all targets that will be deleted.
    ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
        .rule(eventRuleName)
        .build();

    ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
    List<Target> allTargets = response.targets();

    // Get all targets and delete them.
    for (Target myTarget : allTargets) {
        RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
            .rule(eventRuleName)
            .ids(myTarget.id())
            .build();

        eventBrClient.removeTargets(removeTargetsRequest);
        System.out.println("Successfully removed the target");
    }
}
```

- Weitere API-Informationen finden Sie unter [RemoveTargets](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Senden von Ereignissen

Das folgende Codebeispiel zeigt, wie Sie Amazon- EventBridge Ereignisse senden.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
    String json = "{" +
        "\"UserEmail\": \"" + email + "\", " +
        "\"Message\": \"This event was generated by example code.\", " +
        "\"UtcTime\": \"Now.\" " +
        "}";

    PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
        .source("ExampleSource")
        .detail(json)
        .detailType("ExampleType")
        .build();

    PutEventsRequest eventsRequest = PutEventsRequest.builder()
        .entries(entry)
        .build();

    eventBrClient.putEvents(eventsRequest);
}
```

- Weitere API-Informationen finden Sie unter [PutEvents](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Szenarien

### Erste Schritte mit Regeln und Zielen

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 Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
 *
 * This Java V2 example performs the following tasks with Amazon EventBridge:
 *
 * 1. Creates an AWS Identity and Access Management (IAM) role to use with
 * Amazon EventBridge.
 * 2. Amazon Simple Storage Service (Amazon S3) bucket with EventBridge events
 * enabled.
 * 3. Creates a rule that triggers when an object is uploaded to Amazon S3.
 * 4. Lists rules on the event bus.
 * 5. Creates a new Amazon Simple Notification Service (Amazon SNS) topic and
 * lets the user subscribe to it.
 * 6. Adds a target to the rule that sends an email to the specified topic.
 * 7. Creates an EventBridge event that sends an email when an Amazon S3 object
 * is created.
 * 8. Lists Targets.
 * 9. Lists the rules for the same target.
 * 10. Triggers the rule by uploading a file to the Amazon S3 bucket.
 * 11. Disables a specific rule.
 * 12. Checks and print the state of the rule.
 * 13. Adds a transform to the rule to change the text of the email.
 * 14. Enables a specific rule.
```

```

* 15. Triggers the updated rule by uploading a file to the Amazon S3 bucket.
* 16. Updates the rule to be a custom rule pattern.
* 17. Sending an event to trigger the rule.
* 18. Cleans up resources.
*
*/
public class EventbridgeMVP {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException, IOException
    {
        final String usage = ""

            Usage:
                <roleName> <bucketName> <topicName> <eventRuleName>

            Where:
                roleName - The name of the role to create.
                bucketName - The Amazon Simple Storage Service (Amazon S3)
bucket name to create.
                topicName - The name of the Amazon Simple Notification Service
(Amazon SNS) topic to create.
                eventRuleName - The Amazon EventBridge rule name to create.
            """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String polJSON = "{" +
            "\"Version\": \"2012-10-17\", " +
            "\"Statement\": [{" +
            "\"Effect\": \"Allow\", " +
            "\"Principal\": {" +
            "\"Service\": \"events.amazonaws.com\"" +
            "}, " +
            "\"Action\": \"sts:AssumeRole\"" +
            "}] " +
            "}";

        Scanner sc = new Scanner(System.in);
        String roleName = args[0];
        String bucketName = args[1];

```

```
String topicName = args[2];
String eventRuleName = args[3];

Region region = Region.US_EAST_1;
EventBridgeClient eventBrClient = EventBridgeClient.builder()
    .region(region)
    .build();

S3Client s3Client = S3Client.builder()
    .region(region)
    .build();

Region regionGl = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(regionGl)
    .build();

SnsClient snsClient = SnsClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon EventBridge example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out
    .println("1. Create an AWS Identity and Access Management (IAM) role
to use with Amazon EventBridge.");
String roleArn = createIAMRole(iam, roleName, polJSON);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create an S3 bucket with EventBridge events
enabled.");
if (checkBucket(s3Client, bucketName)) {
    System.out.println("Bucket " + bucketName + " already exists. Ending
this scenario.");
    System.exit(1);
}

createBucket(s3Client, bucketName);
Thread.sleep(3000);
setBucketNotification(s3Client, bucketName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a rule that triggers when an object is
uploaded to Amazon S3.");
        Thread.sleep(10000);
        addEventRule(eventBrClient, roleArn, bucketName, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. List rules on the event bus.");
        listRules(eventBrClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create a new SNS topic for testing and let the user
subscribe to the topic.");
        String topicArn = createSnsTopic(snsClient, topicName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add a target to the rule that sends an email to the
specified topic.");
        System.out.println("Enter your email to subscribe to the Amazon SNS
topic:");
        String email = sc.nextLine();
        subEmail(snsClient, topicArn, email);
        System.out.println(
            "Use the link in the email you received to confirm your
subscription. Then, press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create an EventBridge event that sends an email when
an Amazon S3 object is created.");
        addSnsEventRule(eventBrClient, eventRuleName, topicArn, topicName,
eventRuleName, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 8. List Targets.");
        listTargets(eventBrClient, eventRuleName);
        System.out.println(DASHES);
```



```
System.out.println(DASHES);
System.out.println(" 9. List the rules for the same target.");
listTargetRules(eventBrClient, topicArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 10. Trigger the rule by uploading a file to the S3
bucket.");
System.out.println("Press Enter to continue.");
sc.nextLine();
uploadTextFiletoS3(s3Client, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Disable a specific rule.");
changeRuleState(eventBrClient, eventRuleName, false);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Check and print the state of the rule.");
checkRule(eventBrClient, eventRuleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Add a transform to the rule to change the text of
the email.");
updateSnsEventRule(eventBrClient, topicArn, eventRuleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Enable a specific rule.");
changeRuleState(eventBrClient, eventRuleName, true);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 15. Trigger the updated rule by uploading a file to the
S3 bucket.");
System.out.println("Press Enter to continue.");
sc.nextLine();
uploadTextFiletoS3(s3Client, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
```

```

        System.out.println(" 16. Update the rule to be a custom rule pattern.");
        updateToCustomRule(eventBrClient, eventRuleName);
        System.out.println("Updated event rule " + eventRuleName + " to use a custom
pattern.");
        updateCustomRuleTargetWithTransform(eventBrClient, topicArn, eventRuleName);
        System.out.println("Updated event target " + topicArn + ".");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Sending an event to trigger the rule. This will
trigger a subscription email.");
        triggerCustomRule(eventBrClient, email);
        System.out.println("Events have been sent. Press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Clean up resources.");
        System.out.println("Do you want to clean up resources (y/n)");
        String ans = sc.nextLine();
        if (ans.compareTo("y") == 0) {
            cleanupResources(eventBrClient, snsClient, s3Client, iam, topicArn,
eventRuleName, bucketName, roleName);
        } else {
            System.out.println("The resources will not be cleaned up. ");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Amazon EventBridge example scenario has successfully
completed.");
        System.out.println(DASHES);
    }

    public static void cleanupResources(EventBridgeClient eventBrClient, SnsClient
snsClient, S3Client s3Client,
        IamClient iam, String topicArn, String eventRuleName, String bucketName,
String roleName) {
        System.out.println("Removing all targets from the event rule.");
        deleteTargetsFromRule(eventBrClient, eventRuleName);
        deleteRuleByName(eventBrClient, eventRuleName);
        deleteSNSTopic(snsClient, topicArn);
        deleteS3Bucket(s3Client, bucketName);
        deleteRole(iam, roleName);
    }

```

```
}

public static void deleteRole(IamClient iam, String roleName) {
    String policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess";
    DetachRolePolicyRequest policyRequest = DetachRolePolicyRequest.builder()
        .policyArn(policyArn)
        .roleName(roleName)
        .build();

    iam.detachRolePolicy(policyRequest);
    System.out.println("Successfully detached policy " + policyArn + " from role
" + roleName);

    // Delete the role.
    DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
        .roleName(roleName)
        .build();

    iam.deleteRole(roleRequest);
    System.out.println("*** Successfully deleted " + roleName);
}

public static void deleteS3Bucket(S3Client s3Client, String bucketName) {
    // Remove all the objects from the S3 bucket.
    ListObjectsRequest listObjects = ListObjectsRequest.builder()
        .bucket(bucketName)
        .build();

    ListObjectsResponse res = s3Client.listObjects(listObjects);
    List<S3Object> objects = res.contents();
    ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();

    for (S3Object myValue : objects) {
        toDelete.add(ObjectIdentifier.builder()
            .key(myValue.key())
            .build());
    }

    DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
        .bucket(bucketName)
        .delete(Delete.builder()
            .objects(toDelete).build())
        .build();
}
```

```
s3Client.deleteObjects(dor);

// Delete the S3 bucket.
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucketName)
    .build();

s3Client.deleteBucket(deleteBucketRequest);
System.out.println("You have deleted the bucket and the objects");
}

// Delete the SNS topic.
public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
    try {
        DeleteTopicRequest request = DeleteTopicRequest.builder()
            .topicArn(topicArn)
            .build();

        DeleteTopicResponse result = snsClient.deleteTopic(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
    DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
        .name(ruleName)
        .build();

    eventBrClient.deleteRule(ruleRequest);
    System.out.println("Successfully deleted the rule");
}

public static void deleteTargetsFromRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    // First, get all targets that will be deleted.
    ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
        .rule(eventRuleName)
        .build();
```

```
        ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
        List<Target> allTargets = response.targets();

        // Get all targets and delete them.
        for (Target myTarget : allTargets) {
            RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
                .rule(eventRuleName)
                .ids(myTarget.id())
                .build();

            eventBrClient.removeTargets(removeTargetsRequest);
            System.out.println("Successfully removed the target");
        }
    }

    public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
        String json = "{" +
            "\"UserEmail\": \"" + email + "\", " +
            "\"Message\": \"This event was generated by example code.\", " +
            "\"UtcTime\": \"Now.\" " +
            "}";

        PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
            .source("ExampleSource")
            .detail(json)
            .detailType("ExampleType")
            .build();

        PutEventsRequest eventsRequest = PutEventsRequest.builder()
            .entries(entry)
            .build();

        eventBrClient.putEvents(eventsRequest);
    }

    public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
        String ruleName) {
        String targetId = java.util.UUID.randomUUID().toString();
        InputTransformer inputTransformer = InputTransformer.builder()
```

```

        .inputTemplate("\Notification: sample event was received.\")
        .build();

    Target target = Target.builder()
        .id(targetId)
        .arn(topicArn)
        .inputTransformer(inputTransformer)
        .build();

    try {
        PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
            .rule(ruleName)
            .targets(target)
            .eventBusName(null)
            .build();

        eventBrClient.putTargets(targetsRequest);
    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void updateToCustomRule(EventBridgeClient eventBrClient, String
ruleName) {
    String customEventsPattern = "{" +
        "\"source\": [\"ExampleSource\"],\" +
        "\"detail-type\": [\"ExampleType\"]\" +
        "}";

    PutRuleRequest request = PutRuleRequest.builder()
        .name(ruleName)
        .description("Custom test rule")
        .eventPattern(customEventsPattern)
        .build();

    eventBrClient.putRule(request);
}

// Update an Amazon S3 object created rule with a transform on the target.
public static void updateSnsEventRule(EventBridgeClient eventBrClient, String
topicArn, String ruleName) {
    String targetId = java.util.UUID.randomUUID().toString();
    Map<String, String> myMap = new HashMap<>();

```

```
myMap.put("bucket", "$.detail.bucket.name");
myMap.put("time", "$.time");

InputTransformer inputTransformer = InputTransformer.builder()
    .inputTemplate("\\"Notification: an object was uploaded to bucket
<bucket> at <time>.\\"")
    .inputPathsMap(myMap)
    .build();

Target target = Target.builder()
    .id(targetId)
    .arn(topicArn)
    .inputTransformer(inputTransformer)
    .build();

try {
    PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
        .rule(ruleName)
        .targets(target)
        .eventBusName(null)
        .build();

    eventBrClient.putTargets(targetsRequest);

} catch (EventBridgeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

}

public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    try {
        DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
            .name(eventRuleName)
            .build();

        DescribeRuleResponse response = eventBrClient.describeRule(ruleRequest);
        System.out.println("The state of the rule is " +
response.stateAsString());

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
  }  
  
  public static void changeRuleState(EventBridgeClient eventBrClient, String  
eventRuleName, Boolean isEnabled) {  
    try {  
      if (!isEnabled) {  
        System.out.println("Disabling the rule: " + eventRuleName);  
        DisableRuleRequest ruleRequest = DisableRuleRequest.builder()  
          .name(eventRuleName)  
          .build();  
  
        eventBrClient.disableRule(ruleRequest);  
      } else {  
        System.out.println("Enabling the rule: " + eventRuleName);  
        EnableRuleRequest ruleRequest = EnableRuleRequest.builder()  
          .name(eventRuleName)  
          .build();  
        eventBrClient.enableRule(ruleRequest);  
      }  
    } catch (EventBridgeException e) {  
      System.err.println(e.awsErrorDetails().errorMessage());  
      System.exit(1);  
    }  
  }  
  
  // Create and upload a file to an S3 bucket to trigger an event.  
  public static void uploadTextFiletoS3(S3Client s3Client, String bucketName)  
throws IOException {  
    // Create a unique file name.  
    String fileSuffix = new SimpleDateFormat("yyyyMMddHHmmss").format(new  
Date());  
    String fileName = "TextFile" + fileSuffix + ".txt";  
  
    File myFile = new File(fileName);  
    FileWriter fw = new FileWriter(myFile.getAbsolutePath());  
    BufferedWriter bw = new BufferedWriter(fw);  
    bw.write("This is a sample file for testing uploads.");  
    bw.close();  
  
    try {  
      PutObjectRequest putOb = PutObjectRequest.builder()  
        .bucket(bucketName)
```



```
        .key(fileName)
        .build();

        s3Client.putObject(putOb, RequestBody.fromFile(myFile));

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
    ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
        .targetArn(topicArn)
        .build();

    ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
    List<String> rules = response.ruleNames();
    for (String rule : rules) {
        System.out.println("The rule name is " + rule);
    }
}

public static void listTargets(EventBridgeClient eventBrClient, String ruleName)
{
    ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
        .rule(ruleName)
        .build();

    ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
    List<Target> targetsList = res.targets();
    for (Target target: targetsList) {
        System.out.println("Target ARN: "+target.arn());
    }
}

// Add a rule which triggers an SNS target when a file is uploaded to an S3
// bucket.
public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn,
```

```
        String topicName, String eventRuleName, String bucketName) {
    String targetID = java.util.UUID.randomUUID().toString();
    Target myTarget = Target.builder()
        .id(targetID)
        .arn(topicArn)
        .build();

    List<Target> targets = new ArrayList<>();
    targets.add(myTarget);
    PutTargetsRequest request = PutTargetsRequest.builder()
        .eventBusName(null)
        .targets(targets)
        .rule(ruleName)
        .build();

    eventBrClient.putTargets(request);
    System.out.println("Added event rule " + eventRuleName + " with Amazon SNS
target " + topicName + " for bucket "
        + bucketName + ".");
}

public static void subEmail(SnsClient snsClient, String topicArn, String email)
{
    try {
        SubscribeRequest request = SubscribeRequest.builder()
            .protocol("email")
            .endpoint(email)
            .returnSubscriptionArn(true)
            .topicArn(topicArn)
            .build();

        SubscribeResponse result = snsClient.subscribe(request);
        System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
            + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void listRules(EventBridgeClient eventBrClient) {
    try {
```

```

        ListRulesRequest rulesRequest = ListRulesRequest.builder()
            .eventBusName("default")
            .limit(10)
            .build();

        ListRulesResponse response = eventBrClient.listRules(rulesRequest);
        List<Rule> rules = response.rules();
        for (Rule rule : rules) {
            System.out.println("The rule name is : " + rule.name());
            System.out.println("The rule description is : " +
rule.description());
            System.out.println("The rule state is : " + rule.stateAsString());
        }

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String createSnsTopic(SnsClient snsClient, String topicName) {
    String topicPolicy = "{" +
        "\"Version\": \"2012-10-17\", " +
        "\"Statement\": [{" +
        "\"Sid\": \"EventBridgePublishTopic\", " +
        "\"Effect\": \"Allow\", " +
        "\"Principal\": { " +
        "\"Service\": \"events.amazonaws.com\" " +
        "}, " +
        "\"Resource\": \"*\", " +
        "\"Action\": \"sns:Publish\" " +
        "}] " +
        "}";

    Map<String, String> topicAttributes = new HashMap<>();
    topicAttributes.put("Policy", topicPolicy);
    CreateTopicRequest topicRequest = CreateTopicRequest.builder()
        .name(topicName)
        .attributes(topicAttributes)
        .build();

    CreateTopicResponse response = snsClient.createTopic(topicRequest);
    System.out.println("Added topic " + topicName + " for email
subscriptions.");
}

```

```
        return response.topicArn();
    }

    // Create a new event rule that triggers when an Amazon S3 object is created in
    // a bucket.
    public static void addEventRule(EventBridgeClient eventBrClient, String roleArn,
String bucketName,
    String eventRuleName) {
        String pattern = "{\n" +
            "  \"source\": [\"aws.s3\"],\n" +
            "  \"detail-type\": [\"Object Created\"],\n" +
            "  \"detail\": {\n" +
            "    \"bucket\": {\n" +
            "      \"name\": [\"" + bucketName + "\"]}\n" +
            "    }\n" +
            "  }\n" +
            "}";

        try {
            PutRuleRequest ruleRequest = PutRuleRequest.builder()
                .description("Created by using the AWS SDK for Java v2")
                .name(eventRuleName)
                .eventPattern(pattern)
                .roleArn(roleArn)
                .build();

            PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
            System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Determine if the S3 bucket exists.
    public static Boolean checkBucket(S3Client s3Client, String bucketName) {
        try {
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            s3Client.headBucket(headBucketRequest);
        }
    }
}
```

```
        return true;
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    return false;
}

// Set the S3 bucket notification configuration.
public static void setBucketNotification(S3Client s3Client, String bucketName) {
    try {
        EventBridgeConfiguration eventBridgeConfiguration =
EventBridgeConfiguration.builder()
            .build();

        NotificationConfiguration configuration =
NotificationConfiguration.builder()
            .eventBridgeConfiguration(eventBridgeConfiguration)
            .build();

        PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest
            .builder()
            .bucket(bucketName)
            .notificationConfiguration(configuration)
            .skipDestinationValidation(true)
            .build();

        s3Client.putBucketNotificationConfiguration(configurationRequest);
        System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createBucket(S3Client s3Client, String bucketName) {
    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();
    }
}
```

```
s3Client.createBucket(bucketRequest);
HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
    .bucket(bucketName)
    .build();

// Wait until the bucket is created and print out the response.
WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println(bucketName + " is ready");

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

}

public static String createIAMRole(IamClient iam, String rolename, String
polJSON) {
    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(polJSON)
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        AttachRolePolicyRequest rolePolicyRequest =
AttachRolePolicyRequest.builder()
            .roleName(rolename)
            .policyArn("arn:aws:iam::aws:policy/
AmazonEventBridgeFullAccess")
            .build();

        iam.attachRolePolicy(rolePolicyRequest);
        return response.role().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [DeleteRule](#)
  - [DescribeRule](#)
  - [DisableRule](#)
  - [EnableRule](#)
  - [ListRuleNamesByTarget](#)
  - [ListRules](#)
  - [ListTargetsByRule](#)
  - [PutEvents](#)
  - [PutRule](#)
  - [PutTargets](#)

## Prognosebeispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Forecast verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Themen

- [Aktionen](#)

## Aktionen

### Erstellen eines Datensatzes

Das folgende Codebeispiel zeigt, wie Sie einen Forecast-Datensatz erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.CreateDatasetRequest;
import software.amazon.awssdk.services.forecast.model.Schema;
import software.amazon.awssdk.services.forecast.model.SchemaAttribute;
import software.amazon.awssdk.services.forecast.model.CreateDatasetResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateDataSet {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <name>\s

                Where:
                name - The name of the data set.\s
        """;
```



```

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String name = args[0];
    Region region = Region.US_WEST_2;
    ForecastClient forecast = ForecastClient.builder()
        .region(region)
        .build();

    String myDataSetARN = createForecastDataSet(forecast, name);
    System.out.println("The ARN of the new data set is " + myDataSetARN);
    forecast.close();
}

public static String createForecastDataSet(ForecastClient forecast, String name)
{
    try {
        Schema schema = Schema.builder()
            .attributes(getSchema())
            .build();

        CreateDatasetRequest datasetRequest = CreateDatasetRequest.builder()
            .datasetName(name)
            .domain("CUSTOM")
            .datasetType("RELATED_TIME_SERIES")
            .dataFrequency("D")
            .schema(schema)
            .build();

        CreateDatasetResponse response = forecast.createDataset(datasetRequest);
        return response.datasetArn();

    } catch (ForecastException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}

// Create a SchemaAttribute list required to create a data set.
private static List<SchemaAttribute> getSchema() {

```

```
List<SchemaAttribute> schemaList = new ArrayList<>();
SchemaAttribute att1 = SchemaAttribute.builder()
    .attributeName("item_id")
    .attributeType("string")
    .build();

SchemaAttribute att2 = SchemaAttribute.builder()
    .attributeName("timestamp")
    .attributeType("timestamp")
    .build();

SchemaAttribute att3 = SchemaAttribute.builder()
    .attributeName("target_value")
    .attributeType("float")
    .build();

// Push the SchemaAttribute objects to the List.
schemaList.add(att1);
schemaList.add(att2);
schemaList.add(att3);
return schemaList;
}
}
```

- Weitere API-Informationen finden Sie unter [CreateDataset](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen einer Prognose

Das folgende Codebeispiel zeigt, wie Sie eine Prognose erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.CreateForecastRequest;
import software.amazon.awssdk.services.forecast.model.CreateForecastResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateForecast {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <name> <predictorArn>\s

            Where:
                name - The name of the forecast.\s
                predictorArn - The arn of the predictor to use.\s

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        String predictorArn = args[1];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
            .build();

        String forecastArn = createNewForecast(forecast, name, predictorArn);
        System.out.println("The ARN of the new forecast is " + forecastArn);
        forecast.close();
    }
}
```

```
public static String createNewForecast(ForecastClient forecast, String name,
String predictorArn) {
    try {
        CreateForecastRequest forecastRequest = CreateForecastRequest.builder()
            .forecastName(name)
            .predictorArn(predictorArn)
            .build();

        CreateForecastResponse response =
forecast.createForecast(forecastRequest);
        return response.forecastArn();

    } catch (ForecastException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Weitere API-Informationen finden Sie unter [CreateForecast](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen eines Datensatzes

Das folgende Codebeispiel zeigt, wie Sie einen Forecast-Datensatz löschen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DeleteDataset {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <datasetARN>\s

            Where:
                datasetARN - The ARN of the data set to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String datasetARN = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
            .build();

        deleteForecastDataSet(forecast, datasetARN);
        forecast.close();
    }

    public static void deleteForecastDataSet(ForecastClient forecast, String
myDataSetARN) {
        try {
            DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()
                .datasetArn(myDataSetARN)
                .build();

            forecast.deleteDataset(deleteRequest);
            System.out.println("The Data Set was deleted");
        }
    }
}
```

```

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- Weitere API-Informationen finden Sie unter [DeleteDataset](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen einer Prognose

Das folgende Codebeispiel zeigt, wie Sie eine Prognose löschen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDataset {

    public static void main(String[] args) {
        final String usage = ""

```

```

        Usage:
            <datasetARN>\s

        Where:
            datasetARN - The ARN of the data set to delete.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String datasetARN = args[0];
    Region region = Region.US_WEST_2;
    ForecastClient forecast = ForecastClient.builder()
        .region(region)
        .build();

    deleteForecastDataSet(forecast, datasetARN);
    forecast.close();
}

public static void deleteForecastDataSet(ForecastClient forecast, String
myDataSetARN) {
    try {
        DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()
            .datasetArn(myDataSetARN)
            .build();

        forecast.deleteDataset(deleteRequest);
        System.out.println("The Data Set was deleted");

    } catch (ForecastException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Weitere API-Informationen finden Sie unter [DeleteForecast](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beschreiben einer Prognose

Das folgende Codebeispiel zeigt, wie Sie eine Prognoseprognose beschreiben.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DescribeForecastRequest;
import software.amazon.awssdk.services.forecast.model.DescribeForecastResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeForecast {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <forecastarn>\s

                Where:
                forecastarn - The arn of the forecast (for example,
                "arn:aws:forecast:us-west-2:xxxxx322:forecast/my_forecast)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```



```
String forecastarn = args[0];
Region region = Region.US_WEST_2;
ForecastClient forecast = ForecastClient.builder()
    .region(region)
    .build();

describe(forecast, forecastarn);
forecast.close();
}

public static void describe(ForecastClient forecast, String forecastarn) {
    try {
        DescribeForecastRequest request = DescribeForecastRequest.builder()
            .forecastArn(forecastarn)
            .build();

        DescribeForecastResponse response = forecast.describeForecast(request);
        System.out.println("The name of the forecast is " +
response.forecastName());

    } catch (ForecastException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [DescribeForecast](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Datensatzgruppen

Das folgende Codebeispiel zeigt, wie Sie Forecast-Datensatzgruppen auflisten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DatasetGroupSummary;
import software.amazon.awssdk.services.forecast.model.ListDatasetGroupsRequest;
import software.amazon.awssdk.services.forecast.model.ListDatasetGroupsResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListDataSetGroups {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
            .build();

        listDataGroups(forecast);
        forecast.close();
    }

    public static void listDataGroups(ForecastClient forecast) {
        try {
            ListDatasetGroupsRequest group = ListDatasetGroupsRequest.builder()
                .maxResults(10)
                .build();

            ListDatasetGroupsResponse response = forecast.listDatasetGroups(group);
            List<DatasetGroupSummary> groups = response.datasetGroups();
            for (DatasetGroupSummary myGroup : groups) {
                System.out.println("The Data Set name is " +
myGroup.datasetGroupName());
            }
        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```

    }
  }
}

```

- Weitere API-Informationen finden Sie unter [ListDatasetGroups](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Auflisten von Prognosen

Das folgende Codebeispiel zeigt, wie Prognoseprognosen aufgelistet werden.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.ListForecastsResponse;
import software.amazon.awssdk.services.forecast.model.ListForecastsRequest;
import software.amazon.awssdk.services.forecast.model.ForecastSummary;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListForecasts {

    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()

```

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

    listAllForeCasts(forecast);
    forecast.close();
}

public static void listAllForeCasts(ForecastClient forecast) {
    try {
        ListForecastsRequest request = ListForecastsRequest.builder()
            .maxResults(10)
            .build();

        ListForecastsResponse response = forecast.listForecasts(request);
        List<ForecastSummary> forecasts = response.forecasts();
        for (ForecastSummary forecastSummary : forecasts) {
            System.out.println("The name of the forecast is " +
forecastSummary.forecastName());
        }

    } catch (ForecastException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [ListForecasts](#) in der APIAWS SDK for Java 2.x - Referenz für .

## AWS Glue -Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit verwenden AWS Glue.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

## Erste Schritte

### Hallo AWS Glue

Die folgenden Codebeispiele veranschaulichen, wie Sie mit der Verwendung von AWS Glue beginnen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
package com.example.glue;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.ListJobsRequest;
import software.amazon.awssdk.services.glue.model.ListJobsResponse;
import java.util.List;

public class HelloGlue {
    public static void main(String[] args) {
        GlueClient glueClient = GlueClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listJobs(glueClient);
    }

    public static void listJobs(GlueClient glueClient) {
        ListJobsRequest request = ListJobsRequest.builder()
            .maxResults(10)
            .build();
        ListJobsResponse response = glueClient.listJobs(request);
        List<String> jobList = response.jobNames();
        jobList.forEach(job -> {
            System.out.println("Job Name: " + job);
        });
    }
}
```

```
    });  
  }  
}
```

- Weitere API-Informationen finden Sie unter [ListJobs](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Themen

- [Aktionen](#)
- [Szenarien](#)

## Aktionen

### Erstellen eines Crawlers

Das folgende Codebeispiel zeigt, wie Sie einen AWS Glue -Crawler erstellen.

### SDK für Java 2.x

#### Note

Auf [GitHub](#) gibt es mehr. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.glue.GlueClient;  
import software.amazon.awssdk.services.glue.model.CreateCrawlerRequest;  
import software.amazon.awssdk.services.glue.model.CrawlerTargets;  
import software.amazon.awssdk.services.glue.model.GlueException;  
import software.amazon.awssdk.services.glue.model.S3Target;  
import java.util.ArrayList;  
import java.util.List;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic: */
```

```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateCrawler {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <IAM> <s3Path> <cron> <dbName> <crawlerName>

            Where:
                IAM - The ARN of the IAM role that has AWS Glue and S3
permissions.\s
                s3Path - The Amazon Simple Storage Service (Amazon S3) target
that contains data (for example, CSV data).
                cron - A cron expression used to specify the schedule (i.e.,
cron(15 12 * * ? *).
                dbName - The database name.\s
                crawlerName - The name of the crawler.\s
            """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String iam = args[0];
        String s3Path = args[1];
        String cron = args[2];
        String dbName = args[3];
        String crawlerName = args[4];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
            .region(region)
            .build();

        createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
        glueClient.close();
    }

    public static void createGlueCrawler(GlueClient glueClient,
        String iam,
        String s3Path,
        String cron,
```

```

        String dbName,
        String crawlerName) {

    try {
        S3Target s3Target = S3Target.builder()
            .path(s3Path)
            .build();

        // Add the S3Target to a list.
        List<S3Target> targetList = new ArrayList<>();
        targetList.add(s3Target);

        CrawlerTargets targets = CrawlerTargets.builder()
            .s3Targets(targetList)
            .build();

        CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
            .databaseName(dbName)
            .name(crawlerName)
            .description("Created by the AWS Glue Java API")
            .targets(targets)
            .role(iam)
            .schedule(cron)
            .build();

        glueClient.createCrawler(crawlerRequest);
        System.out.println(crawlerName + " was successfully created");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Weitere API-Informationen finden Sie unter [CreateCrawler](#) in der APIAWS SDK for Java 2.x - Referenz für .

Holen Sie sich einen Crawler

Das folgende Codebeispiel zeigt, wie Sie einen AWS Glue -Crawler abrufen.



## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetCrawlerRequest;
import software.amazon.awssdk.services.glue.model.GetCrawlerResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetCrawler {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <crawlerName>

                Where:
                crawlerName - The name of the crawler.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String crawlerName = args[0];
Region region = Region.US_EAST_1;
GlueClient glueClient = GlueClient.builder()
    .region(region)
    .build();

getSpecificCrawler(glueClient, crawlerName);
glueClient.close();
}

public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
{
    try {
        GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
        Instant createDate = response.crawler().creationTime();

        // Convert the Instant to readable date
        DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the Crawler is " + createDate);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [GetCrawler](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Holen Sie sich eine Datenbank aus dem Data Catalog

Das folgende Codebeispiel zeigt, wie Sie eine Datenbank aus der abrufen AWS Glue Data Catalog.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetDatabaseRequest;
import software.amazon.awssdk.services.glue.model.GetDatabaseResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetDatabase {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <databaseName>

                Where:
                databaseName - The name of the database.\s
                """;

        if (args.length != 1) {
```

```

        System.out.println(usage);
        System.exit(1);
    }

    String databaseName = args[0];
    Region region = Region.US_EAST_1;
    GlueClient glueClient = GlueClient.builder()
        .region(region)
        .build();

    getSpecificDatabase(glueClient, databaseName);
    glueClient.close();
}

public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
    try {
        GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
            .name(databaseName)
            .build();

        GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
        Instant createDate = response.database().createTime();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the database is " + createDate);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Weitere API-Informationen finden Sie unter [GetDatabase](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Tabellen aus einer Datenbank abrufen

Das folgende Codebeispiel zeigt, wie Tabellen aus einer Datenbank in abgerufen werden AWS Glue Data Catalog.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetTableRequest;
import software.amazon.awssdk.services.glue.model.GetTableResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetTable {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <dbName> <tableName>

                Where:
                dbName - The database name.\s
                tableName - The name of the table.\s
    }
}
```

```
        """);

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String dbName = args[0];
    String tableName = args[1];
    Region region = Region.US_EAST_1;
    GlueClient glueClient = GlueClient.builder()
        .region(region)
        .build();

    getGlueTable(glueClient, dbName, tableName);
    glueClient.close();
}

public static void getGlueTable(GlueClient glueClient, String dbName, String
tableName) {
    try {
        GetTableRequest tableRequest = GetTableRequest.builder()
            .databaseName(dbName)
            .name(tableName)
            .build();

        GetTableResponse tableResponse = glueClient.getTable(tableRequest);
        Instant createDate = tableResponse.table().createTime();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the table is " + createDate);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [GetTables](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Starten eines Crawlers

Das folgende Codebeispiel zeigt, wie Sie einen AWS Glue -Crawler starten.

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GlueException;
import software.amazon.awssdk.services.glue.model.StartCrawlerRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StartCrawler {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <crawlerName>

                Where:
                crawlerName - The name of the crawler.\s
                """;

        if (args.length != 1) {
```

```
        System.out.println(usage);
        System.exit(1);
    }

    String crawlerName = args[0];
    Region region = Region.US_EAST_1;
    GlueClient glueClient = GlueClient.builder()
        .region(region)
        .build();

    startSpecificCrawler(glueClient, crawlerName);
    glueClient.close();
}

public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
    try {
        StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        glueClient.startCrawler(crawlerRequest);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [StartCrawler](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Szenarien

### Erste Schritte mit Crawlern und Aufträgen

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.



- Auflisten von Informationen zu Datenbanken und Tabellen in Ihrem AWS Glue Data Catalog.
- 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 Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To set up the resources, see this documentation topic:
 *
 * https://docs.aws.amazon.com/glue/latest/ug/tutorial-add-crawler.html
 *
 * This example performs the following tasks:
 *
 * 1. Create a database.
 * 2. Create a crawler.
 * 3. Get a crawler.
 * 4. Start a crawler.
 * 5. Get a database.
 * 6. Get tables.
 * 7. Create a job.
 * 8. Start a job run.
 * 9. List all jobs.
```

```

* 10. Get job runs.
* 11. Delete a job.
* 12. Delete a database.
* 13. Delete a crawler.
*/

```

```

public class GlueScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

            Usage:
                <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName>\s

            Where:
                iam - The ARN of the IAM role that has AWS Glue and S3
permissions.\s
                s3Path - The Amazon Simple Storage Service (Amazon S3) target
that contains data (for example, CSV data).
                cron - A cron expression used to specify the schedule (i.e.,
cron(15 12 * * ? *).
                dbName - The database name.\s
                crawlerName - The name of the crawler.\s
                jobName - The name you assign to this job definition.
                scriptLocation - The Amazon S3 path to a script that runs a job.
                locationUri - The location of the database
                bucketNameSc - The Amazon S3 bucket name used when creating a
job

            """;

        if (args.length != 9) {
            System.out.println(usage);
            System.exit(1);
        }

        String iam = args[0];
        String s3Path = args[1];
        String cron = args[2];
        String dbName = args[3];
        String crawlerName = args[4];
        String jobName = args[5];
        String scriptLocation = args[6];
        String locationUri = args[7];

```

```
String bucketNameSc = args[8];

Region region = Region.US_EAST_1;
GlueClient glueClient = GlueClient.builder()
    .region(region)
    .build();
System.out.println(DASHES);
System.out.println("Welcome to the AWS Glue scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create a database.");
createDatabase(glueClient, dbName, locationUri);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create a crawler.");
createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Get a crawler.");
getSpecificCrawler(glueClient, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Start a crawler.");
startSpecificCrawler(glueClient, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Get a database.");
getSpecificDatabase(glueClient, dbName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("*** Wait 5 min for the tables to become available");
TimeUnit.MINUTES.sleep(5);
System.out.println("6. Get tables.");
String myTableName = getGlueTables(glueClient, dbName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Create a job.");
```

```
        createJob(glueClient, jobName, iam, scriptLocation);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Start a Job run.");
        startJob(glueClient, jobName, dbName, myTableName, bucketNameSc);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List all jobs.");
        getAllJobs(glueClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get job runs.");
        getJobRuns(glueClient, jobName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete a job.");
        deleteJob(glueClient, jobName);
        System.out.println("*** Wait 5 MIN for the " + crawlerName + " to stop");
        TimeUnit.MINUTES.sleep(5);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete a database.");
        deleteDatabase(glueClient, dbName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Delete a crawler.");
        deleteSpecificCrawler(glueClient, crawlerName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Successfully completed the AWS Glue Scenario");
        System.out.println(DASHES);
    }

    public static void createDatabase(GlueClient glueClient, String dbName, String
locationUri) {
        try {
            DatabaseInput input = DatabaseInput.builder()
```

```
        .description("Built with the AWS SDK for Java V2")
        .name(dbName)
        .locationUri(locationUri)
        .build();

    CreateDatabaseRequest request = CreateDatabaseRequest.builder()
        .databaseInput(input)
        .build();

    glueClient.createDatabase(request);
    System.out.println(dbName + " was successfully created");

} catch (GlueException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void createGlueCrawler(GlueClient glueClient,
    String iam,
    String s3Path,
    String cron,
    String dbName,
    String crawlerName) {

    try {
        S3Target s3Target = S3Target.builder()
            .path(s3Path)
            .build();

        List<S3Target> targetList = new ArrayList<>();
        targetList.add(s3Target);
        CrawlerTargets targets = CrawlerTargets.builder()
            .s3Targets(targetList)
            .build();

        CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
            .databaseName(dbName)
            .name(crawlerName)
            .description("Created by the AWS Glue Java API")
            .targets(targets)
            .role(iam)
            .schedule(cron)
            .build();
```

```
        glueClient.createCrawler(crawlerRequest);
        System.out.println(crawlerName + " was successfully created");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
{
    try {
        GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        boolean ready = false;
        while (!ready) {
            GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
            String status = response.crawler().stateAsString();
            if (status.compareTo("READY") == 0) {
                ready = true;
            }
            Thread.sleep(3000);
        }

        System.out.println("The crawler is now ready");

    } catch (GlueException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
    try {
        StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        glueClient.startCrawler(crawlerRequest);
        System.out.println(crawlerName + " was successfully started!");
    }
```

```
    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
    try {
        GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
            .name(databaseName)
            .build();

        GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
        Instant createDate = response.database().createTime();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the database is " + createDate);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String getGlueTables(GlueClient glueClient, String dbName) {
    String myTableName = "";
    try {
        GetTablesRequest tableRequest = GetTablesRequest.builder()
            .databaseName(dbName)
            .build();

        GetTablesResponse response = glueClient.getTables(tableRequest);
        List<Table> tables = response.tableList();
        if (tables.isEmpty()) {
            System.out.println("No tables were returned");
        } else {
```

```
        for (Table table : tables) {
            myTableName = table.name();
            System.out.println("Table name is: " + myTableName);
        }
    }

} catch (GlueException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return myTableName;
}

public static void startJob(GlueClient glueClient, String jobName, String
inputDatabase, String inputTable,
    String outBucket) {
    try {
        Map<String, String> myMap = new HashMap<>();
        myMap.put("--input_database", inputDatabase);
        myMap.put("--input_table", inputTable);
        myMap.put("--output_bucket_url", outBucket);

        StartJobRunRequest runRequest = StartJobRunRequest.builder()
            .workerType(WorkerType.G_1_X)
            .numberOfWorkers(10)
            .arguments(myMap)
            .jobName(jobName)
            .build();

        StartJobRunResponse response = glueClient.startJobRun(runRequest);
        System.out.println("The request Id of the job is " +
response.responseMetadata().requestId());

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createJob(GlueClient glueClient, String jobName, String iam,
String scriptLocation) {
    try {
        JobCommand command = JobCommand.builder()
            .pythonVersion("3")
```



```
        .name("glueetl")
        .scriptLocation(scriptLocation)
        .build();

    CreateJobRequest jobRequest = CreateJobRequest.builder()
        .description("A Job created by using the AWS SDK for Java V2")
        .glueVersion("2.0")
        .workerType(WorkerType.G_1_X)
        .numberOfWorkers(10)
        .name(jobName)
        .role(iam)
        .command(command)
        .build();

    glueClient.createJob(jobRequest);
    System.out.println(jobName + " was successfully created.");

} catch (GlueException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void getAllJobs(GlueClient glueClient) {
    try {
        GetJobsRequest jobsRequest = GetJobsRequest.builder()
            .maxResults(10)
            .build();

        GetJobsResponse jobsResponse = glueClient.getJobs(jobsRequest);
        List<Job> jobs = jobsResponse.jobs();
        for (Job job : jobs) {
            System.out.println("Job name is : " + job.name());
            System.out.println("The job worker type is : " +
job.workerType().name());
        }

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getJobRuns(GlueClient glueClient, String jobName) {
```

```
try {
    GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
        .jobName(jobName)
        .maxResults(20)
        .build();

    boolean jobDone = false;
    while (!jobDone) {
        GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
        List<JobRun> jobRuns = response.jobRuns();
        for (JobRun jobRun : jobRuns) {
            String jobState = jobRun.jobRunState().name();
            if (jobState.compareTo("SUCCEEDED") == 0) {
                System.out.println(jobName + " has succeeded");
                jobDone = true;

            } else if (jobState.compareTo("STOPPED") == 0) {
                System.out.println("Job run has stopped");
                jobDone = true;

            } else if (jobState.compareTo("FAILED") == 0) {
                System.out.println("Job run has failed");
                jobDone = true;

            } else if (jobState.compareTo("TIMEOUT") == 0) {
                System.out.println("Job run has timed out");
                jobDone = true;

            } else {
                System.out.println("*** Job run state is " +
jobRun.jobRunState().name());
                System.out.println("Job run Id is " + jobRun.id());
                System.out.println("The Glue version is " +
jobRun.glueVersion());
            }
            TimeUnit.SECONDS.sleep(5);
        }
    }

} catch (GlueException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

```
public static void deleteJob(GlueClient glueClient, String jobName) {
    try {
        DeleteJobRequest jobRequest = DeleteJobRequest.builder()
            .jobName(jobName)
            .build();

        glueClient.deleteJob(jobRequest);
        System.out.println(jobName + " was successfully deleted");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteDatabase(GlueClient glueClient, String databaseName) {
    try {
        DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
            .name(databaseName)
            .build();

        glueClient.deleteDatabase(request);
        System.out.println(databaseName + " was successfully deleted");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteSpecificCrawler(GlueClient glueClient, String
crawlerName) {
    try {
        DeleteCrawlerRequest deleteCrawlerRequest =
DeleteCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        glueClient.deleteCrawler(deleteCrawlerRequest);
        System.out.println(crawlerName + " was deleted");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
}
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [CreateCrawler](#)
  - [CreateJob](#)
  - [DeleteCrawler](#)
  - [DeleteDatabase](#)
  - [DeleteJob](#)
  - [DeleteTable](#)
  - [GetCrawler](#)
  - [GetDatabase](#)
  - [GetDatabases](#)
  - [GetJob](#)
  - [GetJobRun](#)
  - [GetJobRuns](#)
  - [GetTables](#)
  - [ListJobs](#)
  - [StartCrawler](#)
  - [StartJobRun](#)

## HealthImaging -Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit verwenden HealthImaging.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

## Themen

- [Aktionen](#)
- [Szenarien](#)

## Aktionen

### Hinzufügen eines Tags zu einer Ressource

Das folgende Codebeispiel zeigt, wie Sie einer HealthImaging Ressource ein Tag hinzufügen.

### SDK für Java 2.x

```
public static void tagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
    String resourceArn,
    Map<String, String> tags) {
    try {
        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .resourceArn(resourceArn)
            .tags(tags)
            .build();

        medicalImagingClient.tagResource(tagResourceRequest);

        System.out.println("Tags have been added to the resource.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [TagResource](#) in der APIAWS SDK for Java 2.x - Referenz für .

**Note**

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Kopieren eines Imagesatzes

Das folgende Codebeispiel zeigt, wie Sie einen HealthImaging Bildsatz kopieren.

### SDK für Java 2.x

```
public static String copyMedicalImageSet(MedicalImagingClient
medicalImagingClient,
    String datastoreId,
    String imageSetId,
    String latestVersionId,
    String destinationImageSetId,
    String destinationVersionId) {

    try {
        CopySourceImageSetInformation copySourceImageSetInformation =
CopySourceImageSetInformation.builder()
            .latestVersionId(latestVersionId)
            .build();

        CopyImageSetInformation.Builder copyImageSetBuilder =
CopyImageSetInformation.builder()
            .sourceImageSet(copySourceImageSetInformation);

        if (destinationImageSetId != null) {
            copyImageSetBuilder =
copyImageSetBuilder.destinationImageSet(CopyDestinationImageSet.builder()
                .imageSetId(destinationImageSetId)
                .latestVersionId(destinationVersionId)
                .build());
        }

        CopyImageSetRequest copyImageSetRequest = CopyImageSetRequest.builder()
            .datastoreId(datastoreId)
            .sourceImageSetId(imageSetId)
            .copyImageSetInformation(copyImageSetBuilder.build())
            .build();
```

```

        CopyImageSetResponse response =
medicalImagingClient.copyImageSet(copyImageSetRequest);

        return response.destinationImageSetProperties().imageSetId();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}

```

- Weitere API-Informationen finden Sie unter [CopyImageSet](#) in der APIAWS SDK for Java 2.x - Referenz für .

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Einen Datenspeicher erstellen

Das folgende Codebeispiel zeigt, wie Sie einen HealthImaging Datenspeicher erstellen.

### SDK für Java 2.x

```

public static String createMedicalImageDatastore(MedicalImagingClient
medicalImagingClient,
        String datastoreName) {
    try {
        CreateDatastoreRequest datastoreRequest =
CreateDatastoreRequest.builder()
            .datastoreName(datastoreName)
            .build();

        CreateDatastoreResponse response =
medicalImagingClient.createDatastore(datastoreRequest);
        return response.datastoreId();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

```

```
        System.exit(1);
    }

    return "";
}
```

- Weitere API-Informationen finden Sie unter [CreateDatastore](#) in der APIAWS SDK for Java 2.x - Referenz für .

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Löschen eines Datenspeichers

Das folgende Codebeispiel zeigt, wie Sie einen HealthImaging Datenspeicher löschen.

### SDK für Java 2.x

```
public static void deleteMedicalImagingDatastore(MedicalImagingClient
medicalImagingClient,
        String datastoreID) {
    try {
        DeleteDatastoreRequest datastoreRequest =
DeleteDatastoreRequest.builder()
            .datastoreId(datastoreID)
            .build();
        medicalImagingClient.deleteDatastore(datastoreRequest);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteDatastore](#) in der APIAWS SDK for Java 2.x - Referenz für .



**Note**

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Löschen eines Imagesatzes

Das folgende Codebeispiel zeigt, wie Sie einen HealthImaging Imagesatz löschen.

### SDK für Java 2.x

```
public static void deleteMedicalImageSet(MedicalImagingClient
medicalImagingClient,
    String datastoreId,
    String imagesetId) {
    try {
        DeleteImageSetRequest deleteImageSetRequest =
DeleteImageSetRequest.builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId)
            .build();

        medicalImagingClient.deleteImageSet(deleteImageSetRequest);

        System.out.println("The image set was deleted.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteImageSet](#) in der APIAWS SDK for Java 2.x - Referenz für .

**Note**

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Abrufen eines Image-Frames

Das folgende Codebeispiel zeigt, wie Sie einen Bildrahmen abrufen.

### SDK für Java 2.x

```
public static void getMedicalImageSetFrame(MedicalImagingClient
medicalImagingClient,
        String destinationPath,
        String datastoreId,
        String imagesetId,
        String imageFrameId) {

    try {
        GetImageFrameRequest getImageSetMetadataRequest =
        GetImageFrameRequest.builder()
                                .datastoreId(datastoreId)
                                .imageSetId(imagesetId)
                                .imageFrameInformation(ImageFrameInformation.builder()
                                                        .imageFrameId(imageFrameId)
                                                        .build())
                                .build();

        medicalImagingClient.getImageFrame(getImageSetMetadataRequest,
        FileSystems.getDefault().getPath(destinationPath));

        System.out.println("Image frame downloaded to " +
destinationPath);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [GetImageFrame](#) in der APIAWS SDK for Java 2.x - Referenz für .

**Note**

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Eigenschaften des Datenspeichers abrufen

Das folgende Codebeispiel zeigt, wie Sie HealthImaging Datenspeichereigenschaften abrufen.

### SDK für Java 2.x

```
public static DatastoreProperties getMedicalImageDatastore(MedicalImagingClient
medicalImagingClient,
    String datastoreID) {
    try {
        GetDatastoreRequest datastoreRequest = GetDatastoreRequest.builder()
            .datastoreId(datastoreID)
            .build();
        GetDatastoreResponse response =
medicalImagingClient.getDatastore(datastoreRequest);
        return response.datastoreProperties();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Weitere API-Informationen finden Sie unter [GetDatastore](#) in der APIAWS SDK for Java 2.x - Referenz für .

**Note**

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Abrufen von Imagesatzeigenschaften

Das folgende Codebeispiel zeigt, wie Sie HealthImaging Imagesatzeigenschaften abrufen.

### SDK für Java 2.x

```
public static GetImageSetResponse getMedicalImageSet(MedicalImagingClient
medicalImagingClient,
    String datastoreId,
    String imagesetId,
    String versionId) {
    try {
        GetImageSetRequest.Builder getImageSetRequestBuilder =
        GetImageSetRequest.builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId);

        if (versionId != null) {
            getImageSetRequestBuilder =
            getImageSetRequestBuilder.versionId(versionId);
        }

        return
        medicalImagingClient.getImageSet(getImageSetRequestBuilder.build());
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Weitere API-Informationen finden Sie unter [GetImageSet](#) in der APIAWS SDK for Java 2.x - Referenz für .

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Abrufen von Importauftragseigenschaften

Das folgende Codebeispiel zeigt, wie Sie die Eigenschaften des Importauftrags abrufen.

### SDK für Java 2.x

```
public static DICOMImportJobProperties getDicomImportJob(MedicalImagingClient
medicalImagingClient,
                String datastoreId,
                String jobId) {

    try {
        GetDicomImportJobRequest getDicomImportJobRequest =
        GetDicomImportJobRequest.builder()
            .datastoreId(datastoreId)
            .jobId(jobId)
            .build();

        GetDicomImportJobResponse response =
        medicalImagingClient.getDICOMImportJob(getDicomImportJobRequest);
        return response.jobProperties();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Weitere API-Informationen finden Sie unter [GetDICOMImportJob](#) in der APIAWS SDK for Java 2.x -Referenz für .

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Abrufen von Metadaten für einen Imagesatz

Das folgende Codebeispiel zeigt, wie Metadaten für einen HealthImaging Imagesatz abgerufen werden.

## SDK für Java 2.x

```
public static void getMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
    String destinationPath,
    String datastoreId,
    String imagesetId,
    String versionId) {


    try {
        GetImageSetMetadataRequest.Builder getImageSetMetadataRequestBuilder =
        GetImageSetMetadataRequest.builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId);

        if (versionId != null) {
            getImageSetMetadataRequestBuilder =
            getImageSetMetadataRequestBuilder.versionId(versionId);
        }

        medicalImagingClient.getImageSetMetadata(getImageSetMetadataRequestBuilder.build(),
            FileSystems.getDefault().getPath(destinationPath));

        System.out.println("Metadata downloaded to " + destinationPath);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [GetImageSetMetadata](#) in der APIAWS SDK for Java 2.x -Referenz für .

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Importieren von Massendaten in einen Datenspeicher

Das folgende Codebeispiel zeigt, wie Massendaten in einen HealthImaging Datenspeicher importiert werden.

### SDK für Java 2.x

```
public static String startDicomImportJob(MedicalImagingClient
medicalImagingClient,
    String jobName,
    String datastoreId,
    String dataAccessRoleArn,
    String inputS3Uri,
    String outputS3Uri) {

    try {
        StartDicomImportJobRequest startDicomImportJobRequest =
StartDicomImportJobRequest.builder()
            .jobName(jobName)
            .datastoreId(datastoreId)
            .dataAccessRoleArn(dataAccessRoleArn)
            .inputS3Uri(inputS3Uri)
            .outputS3Uri(outputS3Uri)
            .build();

        StartDicomImportJobResponse response =
medicalImagingClient.startDICOMImportJob(startDicomImportJobRequest);
        return response.jobId();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
```

- Weitere API-Informationen finden Sie unter [StartDICOMImportJob](#) in der APIAWS SDK for Java 2.x -Referenz für .

**Note**

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Datenspeicher auflisten

Das folgende Codebeispiel zeigt, wie Datenspeicher aufgelistet HealthImaging werden.

### SDK für Java 2.x

```
public static List<DatastoreSummary>
listMedicalImagingDatastores(MedicalImagingClient medicalImagingClient) {
    try {
        ListDatastoresRequest datastoreRequest = ListDatastoresRequest.builder()
            .build();
        ListDatastoresIterable responses =
medicalImagingClient.listDatastoresPaginator(datastoreRequest);
        List<DatastoreSummary> datastoreSummaries = new ArrayList<>();

        responses.stream().forEach(response ->
datastoreSummaries.addAll(response.datastoreSummaries()));

        return datastoreSummaries;
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Weitere API-Informationen finden Sie unter [ListDatastores](#) in der APIAWS SDK for Java 2.x - Referenz für .

**Note**

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.



## Auflisten von Imagesatzversionen

Das folgende Codebeispiel zeigt, wie HealthImaging Imagesatz-Versionen aufgelistet werden.

### SDK für Java 2.x

```
public static List<ImageSetProperties>
listMedicalImageSetVersions(MedicalImagingClient medicalImagingClient,
    String datastoreId,
    String imagesetId) {
    try {
        ListImageSetVersionsRequest getImageSetRequest =
ListImageSetVersionsRequest.builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId)
            .build();

        ListImageSetVersionsIterable responses = medicalImagingClient
            .listImageSetVersionsPaginator(getImageSetRequest);
        List<ImageSetProperties> imageSetProperties = new ArrayList<>();
        responses.stream().forEach(response ->
imageSetProperties.addAll(response.imageSetPropertiesList()));

        return imageSetProperties;
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Weitere API-Informationen finden Sie unter [ListImageSetVersions](#) in der APIAWS SDK for Java 2.x -Referenz für .

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Auflisten von Importaufträgen für einen Datenspeicher

Das folgende Codebeispiel zeigt, wie Importaufträge für einen HealthImaging Datenspeicher aufgelistet werden.

### SDK für Java 2.x

```
public static List<DICOMImportJobSummary>
listDicomImportJobs(MedicalImagingClient medicalImagingClient,
                    String datastoreId) {

    try {
        ListDicomImportJobsRequest listDicomImportJobsRequest =
ListDicomImportJobsRequest.builder()
                            .datastoreId(datastoreId)
                            .build();
        ListDicomImportJobsResponse response =
medicalImagingClient.listDICOMImportJobs(listDicomImportJobsRequest);
        return response.jobSummaries();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return new ArrayList<>();
}
```

- Weitere API-Informationen finden Sie unter [ListDICOMImportJobs](#) in der APIAWS SDK for Java 2.x -Referenz für .

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Listet Tags für eine Ressource auf

Das folgende Codebeispiel zeigt, wie Sie Tags für eine HealthImaging Ressource auflisten.


## SDK für Java 2.x

```
public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
    String resourceArn) {
    try {
        ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
            .resourceArn(resourceArn)
            .build();

        return
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Weitere API-Informationen finden Sie unter [ListTagsForResource](#) in der APIAWS SDK for Java 2.x -Referenz für .

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Entfernen eines Tags aus einer Ressource

Das folgende Codebeispiel zeigt, wie Sie ein Tag aus einer HealthImaging Ressource entfernen.

## SDK für Java 2.x

```
public static void untagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
    String resourceArn,
    Collection<String> tagKeys) {
    try {
```

```

        UntagResourceRequest untagResourceRequest =
UntagResourceRequest.builder()
    .resourceArn(resourceArn)
    .tagKeys(tagKeys)
    .build();

        medicalImagingClient.untagResource(untagResourceRequest);

        System.out.println("Tags have been removed from the resource.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Weitere API-Informationen finden Sie unter [UntagResource](#) in der APIAWS SDK for Java 2.x - Referenz für .

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Bildsätze durchsuchen

Das folgende Codebeispiel zeigt, wie Sie HealthImaging Bildsätze durchsuchen.

### SDK für Java 2.x

Die Dienstprogrammfunktion für die Suche nach Bildsätzen.

```

public static List<ImageSetsMetadataSummary> searchMedicalImagingImageSets(
    MedicalImagingClient medicalImagingClient,
    String datastoreId, List<SearchFilter> searchFilters) {
    try {
        SearchImageSetsRequest datastoreRequest =
SearchImageSetsRequest.builder()
            .datastoreId(datastoreId)
            .searchCriteria(SearchCriteria.builder().filters(searchFilters).build())

```

```

        .build());
        SearchImageSetsIterable responses = medicalImagingClient
            .searchImageSetsPaginator(datastoreRequest);
        List<ImageSetsMetadadataSummary> imageSetsMetadadataSummaries =
new ArrayList<>();

        responses.stream().forEach(response ->
imageSetsMetadadataSummaries

.addAll(response.imageSetsMetadadataSummaries()));

        return imageSetsMetadadataSummaries;
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}

```

### Anwendungsfall 1: GLEICHER Operator.

```

        List<SearchFilter> searchFilters =
Collections.singletonList(SearchFilter.builder()
        .operator(Operator.EQUAL)
        .values(SearchByAttributeValue.builder()
            .dicomPatientId(patientId)
            .build())
        .build());

        List<ImageSetsMetadadataSummary> imageSetsMetadadataSummaries =
searchMedicalImagingImageSets(
            medicalImagingClient,
            datastoreId, searchFilters);
        if (imageSetsMetadadataSummaries != null) {
            System.out.println("The image sets for patient " + patientId
+ " are:\n"
                + imageSetsMetadadataSummaries);
            System.out.println();
        }
    }
}

```

## Anwendungsfall 2: BETWEEN-Operator unter Verwendung von Bol StudyDate und Bol StudyTime.

```

        DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("yyyyMMdd");
        searchFilters = Collections.singletonList(SearchFilter.builder()
            .operator(Operator.BETWEEN)
            .values(SearchByAttributeValue.builder()

                .dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()

                    .dicomStudyDate("19990101")

                    .dicomStudyTime("000000.000")

                        .build())

                    .build(),
                    SearchByAttributeValue.builder()

                .dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()

                    .dicomStudyDate((LocalDate.now()

                        .format(formatter)))

                    .dicomStudyTime("000000.000")

                    .build())

                        .build())

                .build());

        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                                datastoreId, searchFilters);
        if (imageSetsMetadataSummaries != null) {
            System.out.println(
                "The image sets searched with BETWEEN
operator using DICOMStudyDate and DICOMStudyTime are:\n"
                    +
                    imageSetsMetadataSummaries);
            System.out.println();
        }

```

### Anwendungsfall 3: BETWEEN-Operator mit createdAt. Zeitreihen wurden zuvor beibehalten.

```

        searchFilters = Collections.singletonList(SearchFilter.builder()
            .operator(Operator.BETWEEN)
            .values(SearchByAttributeValue.builder()

                .createdAt(Instant.parse("1985-04-12T23:20:50.52Z"))
                .build(),
                SearchByAttributeValue.builder()

                .createdAt(Instant.now())
                .build())
            .build());

        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                                datastoreId, searchFilters);
        if (imageSetsMetadataSummaries != null) {
            System.out.println("The image sets searched with BETWEEN
operator using createdAt are:\n "
                                + imageSetsMetadataSummaries);
            System.out.println();
        }

```

- Weitere API-Informationen finden Sie unter [SearchImageSets](#) in der APIAWS SDK for Java 2.x -Referenz für .

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

### Aktualisieren von Imagesatz-Metadaten

Das folgende Codebeispiel zeigt, wie Imagesatz- HealthImaging Metadaten aktualisiert werden.

#### SDK für Java 2.x

```

public static void updateMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,

```

```
        String datastoreId,
        String imagesetId,
        String versionId,
        MetadataUpdates metadataUpdates) {
    try {
        UpdateImageSetMetadataRequest updateImageSetMetadataRequest
= UpdateImageSetMetadataRequest
                                .builder()
                                .datastoreId(datastoreId)
                                .imageSetId(imagesetId)
                                .latestVersionId(versionId)

        .updateImageSetMetadataUpdates(metadataUpdates)
                                .build();

        medicalImagingClient.updateImageSetMetadata(updateImageSetMetadataRequest);

        System.out.println("The image set metadata was updated");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [UpdateImageSetMetadata](#) in der APIAWS SDK for Java 2.x -Referenz für .

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Szenarien

### Markieren eines Datenspeichers

Das folgende Codebeispiel zeigt, wie ein HealthImaging Datenspeicher markiert wird.



## SDK für Java 2.x

So markieren Sie einen Datenspeicher.

```
final String dataStoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

TagResource.tagMedicalImagingResource(medicalImagingClient,
dataStoreArn,
ImmutableMap.of("Deployment", "Development"));
```

Die Hilfsprogrammfunktion zum Markieren einer Ressource.

```
public static void tagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
String resourceArn,
Map<String, String> tags) {
try {
TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
.resourceArn(resourceArn)
.tags(tags)
.build();

medicalImagingClient.tagResource(tagResourceRequest);

System.out.println("Tags have been added to the resource.");
} catch (MedicalImagingException e) {
System.err.println(e.awsErrorDetails().errorMessage());
System.exit(1);
}
}
```

So listen Sie Tags für einen Datenspeicher auf.

```
final String dataStoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

ListTagsForResourceResponse result =
ListTagsForResource.listMedicalImagingResourceTags(
medicalImagingClient,
```

```
        datastoreArn);
    if (result != null) {
        System.out.println("Tags for resource: " + result.tags());
    }
}
```

Die Dienstprogrammfunktion zum Auflisten der Tags einer Ressource.

```
public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
    String resourceArn) {
    try {
        ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
            .resourceArn(resourceArn)
            .build();

        return
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

So heben Sie die Markierung eines Datenspeichers auf.

```
        final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

        UntagResource.untagMedicalImagingResource(medicalImagingClient,
datastoreArn,
            Collections.singletonList("Deployment"));
}
```

Die Hilfsprogrammfunktion zum Aufheben der Markierung einer Ressource.


```
public static void untagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
```

```
        String resourceArn,
        Collection<String> tagKeys) {
    try {
        UntagResourceRequest untagResourceRequest =
UntagResourceRequest.builder()
            .resourceArn(resourceArn)
            .tagKeys(tagKeys)
            .build();

        medicalImagingClient.untagResource(untagResourceRequest);

        System.out.println("Tags have been removed from the resource.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [ListTagsForResource](#)
  - [TagResource](#)
  - [UntagResource](#)

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Markieren eines Imagesatzes

Das folgende Codebeispiel zeigt, wie Sie einen HealthImaging Imagesatz markieren.

### SDK für Java 2.x

So markieren Sie einen Image-Satz.

```
final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
```

```
TagResource.tagMedicalImagingResource(medicalImagingClient,  
imageSetArn,  
ImmutableMap.of("Deployment", "Development"));
```

Die Hilfsprogrammfunktion zum Markieren einer Ressource.

```
public static void tagMedicalImagingResource(MedicalImagingClient  
medicalImagingClient,  
String resourceArn,  
Map<String, String> tags) {  
    try {  
        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()  
            .resourceArn(resourceArn)  
            .tags(tags)  
            .build();  
  
        medicalImagingClient.tagResource(tagResourceRequest);  
  
        System.out.println("Tags have been added to the resource.");  
    } catch (MedicalImagingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

So listen Sie Tags für einen Image-Satz auf.

```
final String imageSetArn = "arn:aws:medical-imaging:us-  
east-1:123456789012: datastore/12345678901234567890123456789012/  
imageset/12345678901234567890123456789012";  
  
ListTagsForResourceResponse result =  
ListTagsForResource.listMedicalImagingResourceTags(  
    medicalImagingClient,  
    imageSetArn);  
if (result != null) {  
    System.out.println("Tags for resource: " + result.tags());  
}
```

## Die Dienstprogrammfunktion zum Auflisten der Tags einer Ressource.

```
public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
    String resourceArn) {
    try {
        ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
            .resourceArn(resourceArn)
            .build();

        return
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

## So heben Sie die Markierung eines Imagesatzes auf.

```
final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012: datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

UntagResource.untagMedicalImagingResource(medicalImagingClient,
    imageSetArn,
        Collections.singletonList("Deployment"));
```

## Die Hilfsprogrammfunktion zum Aufheben der Markierung einer Ressource.

```
public static void untagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
    String resourceArn,
    Collection<String> tagKeys) {
    try {
        UntagResourceRequest untagResourceRequest =
UntagResourceRequest.builder()
            .resourceArn(resourceArn)
```

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

    medicalImagingClient.untagResource(untagResourceRequest);

    System.out.println("Tags have been removed from the resource.");
} catch (MedicalImagingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [ListTagsForResource](#)
  - [TagResource](#)
  - [UntagResource](#)

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## IAM-Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit IAM verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Erste Schritte

## Hallo IAM

Die folgenden Codebeispiele veranschaulichen, wie Sie mit der Verwendung von IAM beginnen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.ListPoliciesResponse;
import software.amazon.awssdk.services.iam.model.Policy;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloIAM {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        listPolicies(iam);
    }

    public static void listPolicies(IamClient iam) {
        ListPoliciesResponse response = iam.listPolicies();
        List<Policy> polList = response.policies();
        polList.forEach(policy -> {
            System.out.println("Policy Name: " + policy.policyName());
        });
    }
}
```

```
}
```

- Weitere API-Informationen finden Sie unter [ListPolicies](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Themen

- [Aktionen](#)
- [Szenarien](#)

## Aktionen

### Anfügen einer Richtlinie an eine Rolle

Das folgende Codebeispiel veranschaulicht, wie Sie eine IAM-Richtlinie an eine Rolle anfügen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```



```
*/
public class AttachRolePolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <roleName> <policyArn>\s

            Where:
                roleName - A role name that you can obtain from the AWS
Management Console.\s
                policyArn - A policy ARN that you can obtain from the AWS
Management Console.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String policyArn = args[1];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        attachIAMRolePolicy(iam, roleName, policyArn);
        iam.close();
    }

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
        try {
            ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                .roleName(roleName)
                .build();

            ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();
        }
    }
}
```

```
// Ensure that the policy is not attached to this role
String polArn = "";
for (AttachedPolicy policy : attachedPolicies) {
    polArn = policy.policyArn();
    if (polArn.compareTo(policyArn) == 0) {
        System.out.println(roleName + " policy is already attached to
this role.");
        return;
    }
}

AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
    .roleName(roleName)
    .policyArn(policyArn)
    .build();

iam.attachRolePolicy(attachRequest);

System.out.println("Successfully attached policy " + policyArn +
    " to role " + roleName);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.println("Done");
}
```

- Weitere API-Informationen finden Sie unter [AttachRolePolicy](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen einer Richtlinie

Das folgende Codebeispiel veranschaulicht, wie Sie eine IAM-Richtlinie erstellen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreatePolicy {

    public static final String PolicyDocument = "{" +
        "  \"Version\": \"2012-10-17\",\" +
        "  \"Statement\": [\" +
        "    {\" +
        "      \"Effect\": \"Allow\",\" +
        "      \"Action\": [\" +
        "        \"dynamodb:DeleteItem\",\" +
        "        \"dynamodb:GetItem\",\" +
        "        \"dynamodb:PutItem\",\" +
        "        \"dynamodb:Scan\",\" +
        "        \"dynamodb:UpdateItem\"" +
        "    ],\" +
        "    \"Resource\": \"*\"" +
        "  }" +
```

```
        "]" +
        "}";

public static void main(String[] args) {

    final String usage = ""
        Usage:
            CreatePolicy <policyName>\s

        Where:
            policyName - A unique policy name.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String policyName = args[0];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    String result = createIAMPolicy(iam, policyName);
    System.out.println("Successfully created a policy with this ARN value: " +
result);
    iam.close();
}

public static String createIAMPolicy(IamClient iam, String policyName) {
    try {
        // Create an IamWaiter object.
        IamWaiter iamWaiter = iam.waiter();

        CreatePolicyRequest request = CreatePolicyRequest.builder()
            .policyName(policyName)
            .policyDocument(PolicyDocument)
            .build();

        CreatePolicyResponse response = iam.createPolicy(request);

        // Wait until the policy is created.
        GetPolicyRequest polRequest = GetPolicyRequest.builder()
```

```
        .policyArn(response.policy().arn())
        .build();

        WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
        return response.policy().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Weitere API-Informationen finden Sie unter [CreatePolicy](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen einer Rolle

Das folgende Codebeispiel veranschaulicht, wie Sie eine IAM-Rolle erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import software.amazon.awssdk.services.iam.model.CreateRoleRequest;
import software.amazon.awssdk.services.iam.model.CreateRoleResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import java.io.FileReader;
```

```
/*
 * This example requires a trust policy document. For more information, see:
 * https://aws.amazon.com/blogs/security/how-to-use-trust-policies-with-iam-roles/
 *
 * In addition, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CreateRole {
    public static void main(String[] args) throws Exception {
        final String usage = ""
            Usage:
                <rolename> <fileLocation>\s

                Where:
                    rolename - The name of the role to create.\s
                    fileLocation - The location of the JSON document that represents
the trust policy.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String rolename = args[0];
        String fileLocation = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        String result = createIAMRole(iam, rolename, fileLocation);
        System.out.println("Successfully created user: " + result);
        iam.close();
    }

    public static String createIAMRole(IamClient iam, String rolename, String
fileLocation) throws Exception {
        try {
```

```
JSONObject jsonObject = (JSONObject) readJsonSimpleDemo(fileLocation);
CreateRoleRequest request = CreateRoleRequest.builder()
    .roleName(rolename)
    .assumeRolePolicyDocument(jsonObject.toJSONString())
    .description("Created using the AWS SDK for Java")
    .build();

CreateRoleResponse response = iam.createRole(request);
System.out.println("The ARN of the role is " + response.role().arn());

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

public static Object readJsonSimpleDemo(String filename) throws Exception {
    FileReader reader = new FileReader(filename);
    JSONParser jsonParser = new JSONParser();
    return jsonParser.parse(reader);
}
}
```

- Weitere API-Informationen finden Sie unter [CreateRole](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen eines Benutzers

Das folgende Codebeispiel veranschaulicht, wie Sie einen IAM-Benutzer erstellen.

### 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](#).

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
import software.amazon.awssdk.services.iam.model.GetUserResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUser {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <username>\s

                Where:
                username - The name of the user to create.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```



```
String username = args[0];
Region region = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(region)
    .build();

String result = createIAMUser(iam, username);
System.out.println("Successfully created user: " + result);
iam.close();
}

public static String createIAMUser(IamClient iam, String username) {
    try {
        // Create an IamWaiter object.
        IamWaiter iamWaiter = iam.waiter();

        CreateUserRequest request = CreateUserRequest.builder()
            .userName(username)
            .build();

        CreateUserResponse response = iam.createUser(request);

        // Wait until the user is created.
        GetUserRequest userRequest = GetUserRequest.builder()
            .userName(response.user().userName())
            .build();

        WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
        waitUntilUserExists.matched().response().ifPresent(System.out::println);
        return response.user().userName();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Weitere API-Informationen finden Sie unter [CreateUser](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen eines Zugriffsschlüssels

Das folgende Codebeispiel veranschaulicht, wie Sie IAM-Zugriffsschlüssel erstellen.

### 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](#).

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.CreateAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAccessKey {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <user>\s
```

```
        Where:
            user - An AWS IAM user that you can obtain from the AWS
Management Console.
        """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String user = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        String keyId = createIAMAccessKey(iam, user);
        System.out.println("The Key Id is " + keyId);
        iam.close();
    }

    public static String createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                .userName(user)
                .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey().accessKeyId();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Weitere API-Informationen finden Sie unter [CreateAccessKey](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erstellen eines Alias für ein Konto

Das folgende Codebeispiel zeigt, wie Sie einen Alias für ein IAM-Konto erstellen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.CreateAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAccountAlias {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <alias>\s

            Where:
                alias - The account alias to create (for example, myawsaccount).

\s

        """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
```

```
IamClient iam = IamClient.builder()
    .region(region)
    .build();

createIAMAccountAlias(iam, alias);
iam.close();
System.out.println("Done");
}

public static void createIAMAccountAlias(IamClient iam, String alias) {
    try {
        CreateAccountAliasRequest request = CreateAccountAliasRequest.builder()
            .accountAlias(alias)
            .build();

        iam.createAccountAlias(request);
        System.out.println("Successfully created account alias: " + alias);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [CreateAccountAlias](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Löschen Sie eine Richtlinie

Das folgende Codebeispiel veranschaulicht, wie Sie eine IAM-Richtlinie löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.DeletePolicyRequest;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeletePolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <policyARN>\s

            Where:
                policyARN - A policy ARN value to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String policyARN = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        deleteIAMPolicy(iam, policyARN);
        iam.close();
    }

    public static void deleteIAMPolicy(IamClient iam, String policyARN) {
        try {
            DeletePolicyRequest request = DeletePolicyRequest.builder()
                .policyArn(policyARN)
                .build();
        }
    }
}
```

```
        iam.deletePolicy(request);
        System.out.println("Successfully deleted the policy");

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
}
```

- Weitere API-Informationen finden Sie unter [DeletePolicy](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen eines Benutzers

Das folgende Codebeispiel veranschaulicht, wie Sie einen IAM-Benutzer löschen.

### 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](#).

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
import software.amazon.awssdk.services.iam.model.IamException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteUser {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <userName>\s

            Where:
                userName - The name of the user to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        deleteIAMUser(iam, userName);
        System.out.println("Done");
        iam.close();
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                .userName(userName)
                .build();

            iam.deleteUser(request);
            System.out.println("Successfully deleted IAM user " + userName);
        }
    }
}
```



```
        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteUser](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen eines Zugriffsschlüssels

Das folgende Codebeispiel veranschaulicht, wie Sie einen IAM-Zugriffsschlüssel löschen.

### 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](#).

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DeleteAccessKey {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <username> <accessKey>\s

            Where:
                username - The name of the user.\s
                accessKey - The access key ID for the secret access key you want
to delete.\s

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        String accessKey = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();
        deleteKey(iam, username, accessKey);
        iam.close();
    }

    public static void deleteKey(IamClient iam, String username, String accessKey) {
        try {
            DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                .accessKeyId(accessKey)
                .userName(username)
                .build();

            iam.deleteAccessKey(request);
            System.out.println("Successfully deleted access key " + accessKey +
                " from user " + username);
        } catch (IamException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [DeleteAccessKey](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen eines Konto-Alias

Das folgende Codebeispiel zeigt, wie Sie einen IAM-Konto-Alias löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.DeleteAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAccountAlias {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <alias>\s
```

```
        Where:
            alias - The account alias to delete.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String alias = args[0];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    deleteIAMAccountAlias(iam, alias);
    iam.close();
}

public static void deleteIAMAccountAlias(IamClient iam, String alias) {
    try {
        DeleteAccountAliasRequest request = DeleteAccountAliasRequest.builder()
            .accountAlias(alias)
            .build();

        iam.deleteAccountAlias(request);
        System.out.println("Successfully deleted account alias " + alias);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
}
```

- Weitere API-Informationen finden Sie unter [DeleteAccountAlias](#) in der APIAWS SDK for Java 2.x -Referenz für .

Trennen Sie eine Richtlinie von einer Rolle

Das folgende Codebeispiel veranschaulicht, wie Sie eine IAM-Richtlinie von einer Rolle trennen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetachRolePolicy {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <roleName> <policyArn>\s

                Where:
                roleName - A role name that you can obtain from the AWS
Management Console.\s
                policyArn - A policy ARN that you can obtain from the AWS
Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String policyArn = args[1];
```

```
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();
        detachPolicy(iam, roleName, policyArn);
        System.out.println("Done");
        iam.close();
    }

    public static void detachPolicy(IamClient iam, String roleName, String
policyArn) {
        try {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(policyArn)
                .build();


            iam.detachRolePolicy(request);
            System.out.println("Successfully detached policy " + policyArn +
                " from role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [DetachRolePolicy](#) in der APIAWS SDK for Java 2.x -Referenz für .

Listen Sie die Zugriffsschlüssel eines Benutzers auf

Das folgende Codebeispiel veranschaulicht, wie Sie die IAM-Zugriffsschlüssel eines Benutzers auflisten.

 **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](#).

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.AccessKeyMetadata;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListAccessKeys {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <userName>\s

                Where:
                userName - The name of the user for which access keys are
retrieved.\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String userName = args[0];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    listKeys(iam, userName);
    System.out.println("Done");
    iam.close();
}

public static void listKeys(IamClient iam, String userName) {
    try {
        boolean done = false;
        String newMarker = null;

        while (!done) {
            ListAccessKeysResponse response;

            if (newMarker == null) {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName)
                    .build();

                response = iam.listAccessKeys(request);
            } else {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName)
                    .marker(newMarker)
                    .build();

                response = iam.listAccessKeys(request);
            }

            for (AccessKeyMetadata metadata : response.accessKeyMetadata()) {
                System.out.format("Retrieved access key %s",
                    metadata.accessKeyId());
            }

            if (!response.isTruncated()) {
                done = true;
            }
        }
    }
}
```



```
        } else {
            newMarker = response.marker();
        }
    }

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [ListAccessKeys](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Konto-Aliasnamen

Das folgende Codebeispiel zeigt, wie Sie IAM-Konto-Aliase auflisten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccountAliasesResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class ListAccountAliases {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        listAliases(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAliases(IamClient iam) {
        try {
            ListAccountAliasesResponse response = iam.listAccountAliases();
            for (String alias : response.accountAliases()) {
                System.out.printf("Retrieved account alias %s", alias);
            }
        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [ListAccountAliases](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Auflisten von Benutzern

Das folgende Codebeispiel veranschaulicht, wie Sie IAM-Benutzer auflisten.

### 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](#).

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.AttachedPermissionsBoundary;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.User;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUsers {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        listAllUsers(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAllUsers(IamClient iam) {
        try {
            boolean done = false;
            String newMarker = null;
            while (!done) {
                ListUsersResponse response;
```

```
        if (newMarker == null) {
            ListUsersRequest request = ListUsersRequest.builder().build();
            response = iam.listUsers(request);
        } else {
            ListUsersRequest request = ListUsersRequest.builder()
                .marker(newMarker)
                .build();

            response = iam.listUsers(request);
        }

        for (User user : response.users()) {
            System.out.format("\n Retrieved user %s", user.userName());
            AttachedPermissionsBoundary permissionsBoundary =
user.permissionsBoundary();
            if (permissionsBoundary != null)
                System.out.format("\n Permissions boundary details %s",
permissionsBoundary.permissionsBoundaryTypeAsString());
        }

        if (!response.isTruncated()) {
            done = true;
        } else {
            newMarker = response.marker();
        }
    }

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [ListUsers](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Aktualisieren eines Benutzers

Das folgende Codebeispiel zeigt, wie Sie einen IAM-Benutzer aktualisieren.

**⚠ 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](#).

## SDK für Java 2.x

**ℹ Note**

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateUser {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <curName> <newName>\s

                Where:
                curName - The current user name.\s
                newName - An updated user name.\s
        """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String curName = args[0];
        String newName = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        updateIAMUser(iam, curName, newName);
        System.out.println("Done");
        iam.close();
    }

    public static void updateIAMUser(IamClient iam, String curName, String newName)
    {
        try {
            UpdateUserRequest request = UpdateUserRequest.builder()
                .userName(curName)
                .newUserName(newName)
                .build();

            iam.updateUser(request);
            System.out.printf("Successfully updated user to username %s", newName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [UpdateUser](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Aktualisieren eines Zugriffsschlüssels

Das folgende Codebeispiel zeigt, wie Sie einen IAM-Zugriffsschlüssel aktualisieren.

**⚠ 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](#).

## SDK für Java 2.x

**ℹ Note**

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.StatusType;
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateAccessKey {

    private static StatusType statusType;

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <username> <accessId> <status>\s

        Where:
```

```
        username - The name of the user whose key you want to update.\s
        accessId - The access key ID of the secret access key you want
to update.\s
        status - The status you want to assign to the secret access key.
\s
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String username = args[0];
    String accessId = args[1];
    String status = args[2];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    updateKey(iam, username, accessId, status);
    System.out.println("Done");
    iam.close();
}

public static void updateKey(IamClient iam, String username, String accessId,
String status) {
    try {
        if (status.toLowerCase().equalsIgnoreCase("active")) {
            statusType = StatusType.ACTIVE;
        } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
            statusType = StatusType.INACTIVE;
        } else {
            statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
        }

        UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
            .accessKeyId(accessId)
            .userName(username)
            .status(statusType)
            .build();

        iam.updateAccessKey(request);
    }
}
```



```
        System.out.printf("Successfully updated the status of access key %s to"
+
        "status %s for user %s", accessId, status, username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [UpdateAccessKey](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Szenarien

### Erstellen und Verwalten eines ausfallsicheren Services

Das folgende Codebeispiel zeigt, wie Sie einen Webservice mit Load Balancing erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Antwort des Webserver auf Anfragen und Zustandsprüfungen, indem Sie die AWS Systems Manager Parameter aktualisieren.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\\
\\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\\
\\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\\
\\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\\
\\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\\0", "-");
```

```
public static void main(String[] args) throws IOException, InterruptedException
{
    Scanner in = new Scanner(System.in);
    Database database = new Database();
    AutoScaler autoScaler = new AutoScaler();
    LoadBalancer loadBalancer = new LoadBalancer();

    System.out.println(DASHES);
    System.out.println("Welcome to the demonstration of How to Build and Manage
a Resilient Service!");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("A - SETUP THE RESOURCES");
    System.out.println("Press Enter when you're ready to start deploying
resources.");
    in.nextLine();
    deploy(loadBalancer);
    System.out.println(DASHES);
    System.out.println(DASHES);
    System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    demo(loadBalancer);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("C - DELETE THE RESOURCES");
    System.out.println("""
        This concludes the demo of how to build and manage a resilient
service.
        To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
        that were created for this demo.
        """);

    System.out.println("\n Do you want to delete the resources (y/n)? ");
    String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

    if (userInput.equals("y")) {
        // Delete resources here
        deleteResources(loadBalancer, autoScaler, database);
        System.out.println("Resources deleted.");
    } else {
```

```

        System.out.println("""
            Okay, we'll leave the resources intact.
            Don't forget to delete them when you're done with them or you
might incur unexpected charges.
            """);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("The example has completed. ");
    System.out.println("\n Thanks for watching!");
    System.out.println(DASHES);
}

// Deletes the AWS resources used in this example.
private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
    throws IOException, InterruptedException {
    loadBalancer.deleteLoadBalancer(lbName);
    System.out.println("*** Wait 30 secs for resource to be deleted");
    TimeUnit.SECONDS.sleep(30);
    loadBalancer.deleteTargetGroup(targetGroupName);
    autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
    autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
    autoScaler.deleteTemplate(templateName);
    database.deleteTable(tableName);
}

private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
    Scanner in = new Scanner(System.in);
    System.out.println(
        """

            For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources

            to set up a load-balanced web service endpoint and explore
some ways to make it resilient

            against various kinds of failures.

            Some of the resources create by this demo are:
            \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
            \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.

```

```
        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
        """);

    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Creating and populating a DynamoDB table named " +
tableName);
    Database database = new Database();
    database.createTable(tableName, fileName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("""
        Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
        This script starts a Python web server defined in the `server.py`
script. The web server
        listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
        For demo purposes, this server is run as the root user. In
production, the best practice is to
        run a web server, such as Apache, with least-privileged credentials.

        The template also defines an IAM policy that each instance uses to
assume a role that grants
        permissions to access the DynamoDB recommendation table and Systems
Manager parameters
        that control the flow of the demo.
        """);

    LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
    templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println(
```

```
        "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
            At this point, you have EC2 instances created. Once each instance
starts, it listens for
            HTTP requests. You can see these instances in the console or
continue with the demo.
            Press Enter when you're ready to continue.
            """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
            Creating an Elastic Load Balancing target group and load balancer.
The target group
            defines how the load balancer connects to instances. The load
balancer provides a
            single endpoint where clients connect and dispatches requests to
instances in the group.
            """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
```

```

        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessful) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                        For this example to work, the default security group for
your default VPC must
                        allow access from this computer. You can either add it
automatically from this
                        example or add it yourself using the AWS Management
Console.
                        """);

                    System.out.println(
                        "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {

```

```

        System.out.println("No security group rule added.");
    }
}

} catch (AutoScalingException e) {
    e.printStackTrace();
}
} else if (wasSuccessful) {
    System.out.println("Your load balancer is ready. You can access it by
browsing to:");
    System.out.println("\t http://" + elbDnsName);
} else {
    System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
    System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
    System.out.println("you can successfully make a GET request to the load
balancer.");
}

    System.out.println("Press Enter when you're ready to continue with the
demo.");
    in.nextLine();
}

// A method that controls the demo part of the Java program.
public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    ParameterHelper paramHelper = new ParameterHelper();
    System.out.println("Read the ssm_only_policy.json file");
    String ssmOnlyPolicy = readFileAsString(ssmJSON);

    System.out.println("Resetting parameters to starting values for demo.");
    paramHelper.reset();

    System.out.println(
        """"
                This part of the demonstration shows how to toggle
different parts of the system
                to create situations where the web service fails, and shows
how using a resilient
                architecture can keep the web service running in spite of
these failures.
        """);
}

```



```
        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
        """);
        demoChoices(loadBalancer);

        System.out.println(
            ""
                The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                The table name is contained in a Systems Manager parameter
named self.param_helper.table.
                To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
            """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
            ""
                \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
                healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
            """);
        demoChoices(loadBalancer);

        System.out.println(
            ""
                Instead of failing when the recommendation service fails,
the web service can return a static response.
                While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
            """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
            Now, sending a GET request to the load balancer endpoint returns a
static response.
            The service still reports as healthy because health checks are still
shallow.
            """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);
```

```
System.out.println("""
    Let's also substitute bad credentials for one of the instances in
the target group so that it can't
    access the DynamoDB recommendation table. We will get an instance id
value.
    """);

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
AutoScaler autoScaler = new AutoScaler();

// Create a new instance profile based on badCredsProfileName.
templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
System.out.println("The association Id value is " + profileAssociationId);
System.out.println("Replacing the profile for instance " + badInstanceId
    + " with a profile that contains bad credentials");
autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

System.out.println(
    """
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
        depending on which instance is selected by the load
balancer.
    """);

demoChoices(loadBalancer);

System.out.println("""
    Let's implement a deep health check. For this demo, a deep health
check tests whether
    the web service can access the DynamoDB table that it depends on for
recommendations. Note that
    the deep health check is only for ELB routing and not for Auto
Scaling instance health.
    This kind of deep health check is not recommended for Auto Scaling
instance health, because it
```

```
        risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
        """);

        System.out.println("""
        By implementing deep health checks, the load balancer can detect
when one of the instances is failing
        and take that instance out of rotation.
        """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
        Now, checking target health indicates that the instance with bad
credentials
        is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
        instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
        the load balancer takes unhealthy instances out of its rotation.
        """);

        demoChoices(loadBalancer);

        System.out.println(
        ""
                Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                instance is to terminate it and let the auto scaler start a
new instance to replace it.
        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
        Even while the instance is terminating and the new instance is
starting, sending a GET
        request to the web service continues to get a successful
recommendation response because
        the load balancer routes requests to the healthy instances. After
the replacement instance
        starts and reports as healthy, it is included in the load balancing
rotation.

        Note that terminating and replacing an instance typically takes
several minutes, during which time you
```

can see the changing health check status until the new instance is running and healthy.

```
        """);

        demoChoices(loadBalancer);
        System.out.println(
            "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
    InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClient.createDefault();
```

```
        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

        // Execute the request and get the response.
        HttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        System.out.println("HTTP Status Code: " + statusCode);

        // Display the JSON response
        BufferedReader reader = new BufferedReader(
            new
InputStreamReader(response.getEntity().getContent()));
        StringBuilder jsonResponse = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            jsonResponse.append(line);
        }
        reader.close();

        // Print the formatted JSON response.
        System.out.println("Full Response:\n");
        System.out.println(jsonResponse.toString());

        // Close the HTTP client.
        httpClient.close();
    }
    case 1 -> {
        System.out.println("\nChecking the health of load balancer
targets:\n");

        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println("""
            Note that it can take a minute or two for the health
check to update

            after changes are made.
            """);
    }
}
```

```

        }
        case 2 -> {
            System.out.println("\n0kay, let's move on.");
            System.out.println("-".repeat(88));
            return; // Exit the method when choice is 2
        }
        default -> System.out.println("You must choose a value between
0-2. Please select again.");
    }

    } catch (java.util.InputMismatchException e) {
        System.out.println("Invalid input. Please select again.");
        scanner.nextLine(); // Clear the input buffer.
    }
}

}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}

```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```

public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return iamClient;
    }
}

```

```
private SsmClient getSSMClient() {
    if (ssmClient == null) {
        ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ssmClient;
}

private Ec2Client getEc2Client() {
    if (ec2Client == null) {
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance is
 * terminated, it can no longer be accessed.
 */
public void terminateInstance(String instanceId) {
    TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
    TerminateInstanceInAutoScalingGroupRequest
        .builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

    getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
    System.out.format("Terminated instance %s.", instanceId);
}
}
```

```
/**
 * Replaces the profile associated with a running instance. After the profile is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
 * When
 * the instance is ready, Systems Manager is used to restart the Python web
 * server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
    throws InterruptedException {
    // Create an IAM instance profile specification.
    software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
    .builder()
    .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
    .build();

    // Replace the IAM instance profile association for the EC2 instance.
    ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
    .builder()
    .iamInstanceProfile(iamInstanceProfile)
    .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
    .build();

    try {
        getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
        // Handle the response as needed.
    } catch (Ec2Exception e) {
        // Handle exceptions, log, or report the error.
        System.err.println("Error: " + e.getMessage());
    }

    System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
        newInstanceProfileName);
    TimeUnit.SECONDS.sleep(15);
    boolean instReady = false;
    int tries = 0;

    // Reboot after 60 seconds
```



```
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                    .instanceIds(instanceId)
                    .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
            .instanceIds(instanceId)
            .documentName("AWS-RunShellScript")
            .parameters(Collections.singletonMap("commands",
                Collections.singletonList("cd / && sudo python3 server.py
80")))
            .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(secGroupId)
```

```
        .cidrIp(ipAddress)
        .fromPort(Integer.parseInt(port))
        .build();

    getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
    System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
}

/**
 * Detaches a role from an instance profile, detaches policies from the role,
 * and deletes all the resources.
 */
public void deleteInstanceProfile(String roleName, String profileName) {
    try {
        software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
        .builder()
        .instanceProfileName(profileName)
        .build();

        GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
        String name = response.getInstanceProfile().getInstanceProfileName();
        System.out.println(name);

        RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(profileName)
        .roleName(roleName)
        .build();

        getIAMClient().removeRoleFromInstanceProfile(profileRequest);
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
        .instanceProfileName(profileName)
        .build();

        getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
        System.out.println("Deleted instance profile " + profileName);

        DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
        .roleName(roleName)
```

```
        .build();

        // List attached role policies.
        ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
            .listAttachedRolePolicies(role -> role.roleName(roleName));
        List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
        for (AttachedPolicy attachedPolicy : attachedPolicies) {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(attachedPolicy.policyArn())
                .build();

            getIAMClient().detachRolePolicy(request);
            System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
        }

        getIAMClient().deleteRole(deleteRoleRequest);
        System.out.println("Instance profile and role deleted.");

    } catch (IamException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
    System.out.format(templateName + " was deleted.");
}

public void deleteAutoScaleGroup(String groupName) {
    DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
        .autoScalingGroupName(groupName)
        .forceDelete(true)
        .build();

    getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println(groupName + " was deleted.");
}
```

```
/*
 * Verify the default security group of the specified VPC allows ingress from
 * this
 * computer. This can be done by allowing ingress from this computer's IP
 * address. In some situations, such as connecting from a corporate network, you
 * must instead specify a prefix list ID. You can also temporarily open the port
 * to
 * any IP address while running this example. If you do, be sure to remove
 * public
 * access when you're done.
 *
 */
public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
    boolean portIsOpen = false;
    GroupInfo groupInfo = new GroupInfo();
    try {
        Filter filter = Filter.builder()
            .name("group-name")
            .values("default")
            .build();

        Filter filter1 = Filter.builder()
            .name("vpc-id")
            .values(VPC)
            .build();

        DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
            .filters(filter, filter1)
            .build();

        DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
            .describeSecurityGroups(securityGroupsRequest);
        String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
        groupInfo.setGroupName(securityGroup);

        for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
            System.out.println("Found security group: " + secGroup.groupId());

            for (IpPermission ipPermission : secGroup.ipPermissions()) {
                if (ipPermission.fromPort() == port) {
                    System.out.println("Found inbound rule: " + ipPermission);
                }
            }
        }
    }
}
```

```

        for (IpRange ipRange : ipPermission.ipRanges()) {
            String cidrIp = ipRange.cidrIp();
            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                System.out.println(cidrIp + " is applicable");
                portIsOpen = true;
            }
        }

        if (!ipPermission.prefixListIds().isEmpty()) {
            System.out.println("Prefix lList is applicable");
            portIsOpen = true;
        }

        if (!portIsOpen) {
            System.out
                .println("The inbound rule does not appear to be
open to either this computer's IP,"
                        + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
        } else {
            break;
        }
    }
}

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}

groupInfo.setPortOpen(portIsOpen);
return groupInfo;
}

/*
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {

```

```
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                .autoScalingGroupName(asGroupName)
                .targetGroupARNs(targetGroupARN)
                .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.
        software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
```

```
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
    .launchTemplate(specification)
    .availabilityZones(zones)
    .maxSize(groupSize)
    .minSize(groupSize)
    .autoScalingGroupName(autoScalingGroupName)
    .build();

    try {
        getAutoScalingClient().createAutoScalingGroup(groupRequest);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
    return zones;
}

public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();

    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
        .builder()
        .filters(defaultFilter)
        .build();

    DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
    return response.vpcs().get(0).vpcId();
}

// Gets the default subnets in a VPC for a specified list of Availability Zones.
public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
    List<Subnet> subnets = null;
    Filter vpcFilter = Filter.builder()
        .name("vpc-id")
        .values(vpcId)
```

```
        .build();

    Filter azFilter = Filter.builder()
        .name("availability-zone")
        .values(availabilityZones)
        .build();

    Filter defaultForAZ = Filter.builder()
        .name("default-for-az")
        .values("true")
        .build();

    DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
        .filters(vpcFilter, azFilter, defaultForAZ)
        .build();

    DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
    subnets = response.subnets();
    return subnets;
}

// Gets data about the instances in the EC2 Auto Scaling group.
public String getBadInstance(String groupName) {
    DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

    DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
    AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
    List<String> instanceIds = autoScalingGroup.instances().stream()
        .map(instance -> instance.instanceId())
        .collect(Collectors.toList());

    String[] instanceIdArray = instanceIds.toArray(new String[0]);
    for (String instanceId : instanceIdArray) {
        System.out.println("Instance ID: " + instanceId);
        return instanceId;
    }
    return "";
}

// Gets data about the profile associated with an instance.
```



```
public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();

    DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
        .describeIamInstanceProfileAssociations(associationsRequest);
    return response.iamInstanceProfileAssociations().get(0).associationId();
}

public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
    ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
    ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
    for (Policy policy : listPoliciesResponse.policies()) {
        if (policy.policyName().equals(policyName)) {
            // List the entities (users, groups, roles) that are attached to the
policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
        .builder()
        .policyArn(policy.arn())
        .build();
        ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
            .listEntitiesForPolicy(listEntitiesRequest);
        if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
            || !listEntitiesResponse.policyRoles().isEmpty()) {
            // Detach the policy from any entities it is attached to.
            DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
                .policyArn(policy.arn())
                .roleName(roleName) // Specify the name of the IAM role
```

```
        .build();

        getIAMClient().detachRolePolicy(detachPolicyRequest);
        System.out.println("Policy detached from entities.");
    }

    // Now, you can delete the policy.
    DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
        .policyArn(policy.arn())
        .build();

    getIAMClient().deletePolicy(deletePolicyRequest);
    System.out.println("Policy deleted successfully.");
    break;
}
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
    RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(InstanceProfile)
        .roleName(roleName) // Remove the extra dot here
        .build();

    getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
    System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
}

// Delete the instance profile after removing all roles
DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
    .instanceProfileName(InstanceProfile)
```

```

        .build();

        getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}

```

Erstellen Sie eine Klasse, die Elastic-Load-Balancing-Aktionen beinhaltet.

```

public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
            .names(targetGroupName)
            .build();

        DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
            .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
            .build();

        DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }
}

```

```
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName) {
    DescribeLoadBalancersResponse res = getLoadBalancerClient()
        .describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
```

```

        .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to the ELB.
    HttpGet httpGet = new HttpGet("http://" + elbDnsName);
    try {
        while ((!success) && (retries > 0)) {
            // Execute the request and get the response.
            HttpResponse response = httpClient.execute(httpGet);
            int statusCode = response.getStatusLine().getStatusCode();
            System.out.println("HTTP Status Code: " + statusCode);
            if (statusCode == 200) {
                success = true;
            } else {
                retries--;
                System.out.println("Got connection error from load balancer
endpoint, retrying...");
                TimeUnit.SECONDS.sleep(15);
            }
        }

    } catch (org.apache.http.conn.HttpHostConnectException e) {
        System.out.println(e.getMessage());
    }

    System.out.println("Status.." + success);
    return success;
}

/*
 * Creates an Elastic Load Balancing target group. The target group specifies

```

```
    * how
    * the load balancer forward requests to instances in the group and how instance
    * health is checked.
    */
    public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
            .healthCheckPath("/healthcheck")
            .healthCheckTimeoutSeconds(5)
            .port(port)
            .vpcId(vpcId)
            .name(targetGroupName)
            .protocol(protocol)
            .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /**
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
        String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                .map(Subnet::subnetId)
                .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                .subnets(subnetIdStrings)
                .name(lbName)
```

```
        .scheme("internet-facing")
        .build();

    // Create and wait for the load balancer to become available.
    CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
    String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

    ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
    DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
        .loadBalancerArns(lbARN)
        .build();

    System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
    WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
        .waitUntilLoadBalancerAvailable(request);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Load Balancer " + lbName + " is available.");

    // Get the DNS name (endpoint) of the load balancer.
    String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
    System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

    // Create a listener for the load balance.
    Action action = Action.builder()
        .targetGroupArn(targetGroupARN)
        .type("forward")
        .build();

    CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
        .defaultActions(action)
        .port(port)
        .protocol(protocol)
        .defaultActions(action)
        .build();

    getLoadBalancerClient().createListener(listenerRequest);
```

```

        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}

```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```

public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;
        }
    }
}

```



```
    } catch (ResourceNotFoundException e) {
        System.out.println("Table '" + tableName + "' does not exist.");
    } catch (DynamoDbException e) {
        System.err.println("Error checking table existence: " + e.getMessage());
    }
    return false;
}

/**
 * Creates a DynamoDB table to use a recommendation service. The table has a
 * hash key named 'MediaType' that defines the type of media recommended, such
 * as
 * Book or Movie, and a range key named 'ItemId' that, combined with the
 * MediaType,
 * forms a unique identifier for the recommended item.
 */
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("ItemId")
                    .attributeType(ScalarAttributeType.N)
                    .build())
            .keySchema(
                KeySchemaElement.builder()
                    .attributeName("MediaType")
                    .keyType(KeyType.HASH)
                    .build(),
                KeySchemaElement.builder()
                    .attributeName("ItemId")
                    .keyType(KeyType.RANGE)
                    .build())
            .provisionedThroughput(
                ProvisionedThroughput.builder()
                    .readCapacityUnits(5L)
    }
```

```

        .writeCapacityUnits(5L)
        .build())
        .build();

    getDynamoDbClient().createTable(createTableRequest);
    System.out.println("Creating table " + tableName + "...");

    // Wait until the Amazon DynamoDB table is created.
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    WaiterResponse<DescribeTableResponse> waiterResponse =
    dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Table " + tableName + " created.");

    // Add records to the table.
    populateTable(fileName, tableName);
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.
public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
    }
}

```

```
String creator = currentNode.path("Creator").path("S").asText();

// Create a Recommendation object and set its properties.
Recommendation rec = new Recommendation();
rec.setMediaType(mediaType);
rec.setItemId(itemId);
rec.setTitle(title);
rec.setCreator(creator);

// Put the item into the DynamoDB table.
mappedTable.putItem(rec); // Add the Recommendation to the list.
}
System.out.println("Added all records to the " + tableName);
}
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();


        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
            .overwrite(true)
            .type("String")
            .build();
```

```
        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [AttachLoadBalancerTargetGroups](#)
  - [CreateAutoScalingGroup](#)
  - [CreateInstanceProfile](#)
  - [CreateLaunchTemplate](#)
  - [CreateListener](#)
  - [CreateLoadBalancer](#)
  - [CreateTargetGroup](#)
  - [DeleteAutoScalingGroup](#)
  - [DeleteInstanceProfile](#)
  - [DeleteLaunchTemplate](#)
  - [DeleteLoadBalancer](#)
  - [DeleteTargetGroup](#)
  - [DescribeAutoScalingGroups](#)
  - [DescribeAvailabilityZones](#)
  - [DescribeIamInstanceProfileAssociations](#)
  - [DescribeInstances](#)
  - [DescribeLoadBalancers](#)
  - [DescribeSubnets](#)
  - [DescribeTargetGroups](#)
  - [DescribeTargetHealth](#)
  - [DescribeVpcs](#)
  - [RebootInstances](#)
  - [ReplacelamInstanceProfileAssociation](#)
  - [TerminateInstanceInAutoScalingGroup](#)
  - [UpdateAutoScalingGroup](#)

Erstellen Sie einen Benutzer und nehmen Sie eine Rolle an


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 Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Erstellen Sie Funktionen, die IAM-Benutzer-Aktionen umschließen.

```
/*  
  To run this Java V2 code example, set up your development environment, including  
  your credentials.  
  
  For information, see this documentation topic:  
  
  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
  
  This example performs these operations:
```

1. Creates a user that has no permissions.
  2. Creates a role and policy that grants Amazon S3 permissions.
  3. Creates a role.
  4. Grants the user permissions.
  5. Gets temporary credentials by assuming the role. Creates an Amazon S3 Service client object with the temporary credentials.
  6. Deletes the resources.
- \*/

```
public class IAMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static final String PolicyDocument = "{" +
        "  \"Version\": \"2012-10-17\"," +
        "  \"Statement\": [" +
        "    {" +
        "      \"Effect\": \"Allow\"," +
        "      \"Action\": [" +
        "        \"s3:*\"" +
        "      ]," +
        "      \"Resource\": \"*\\"" +
        "    }" +
        "  ]" +
        "};

    public static String userArn;

    public static void main(String[] args) throws Exception {

        final String usage = ""

            Usage:
              <username> <policyName> <roleName> <roleSessionName>
<bucketName>\s

            Where:
              username - The name of the IAM user to create.\s
              policyName - The name of the policy to create.\s
              roleName - The name of the role to create.\s
              roleSessionName - The name of the session required for the
assumeRole operation.\s
              bucketName - The name of the Amazon S3 bucket from which objects
are read.\s

            """;
```

```
if (args.length != 5) {
    System.out.println(usage);
    System.exit(1);
}

String userName = args[0];
String policyName = args[1];
String roleName = args[2];
String roleSessionName = args[3];
String bucketName = args[4];

Region region = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the AWS IAM example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 1. Create the IAM user.");
User createUser = createIAMUser(iam, userName);

System.out.println(DASHES);
userArn = createUser.arn();

AccessKey myKey = createIAMAccessKey(iam, userName);
String accessKey = myKey.accessKeyId();
String secretKey = myKey.secretAccessKey();
String assumeRolePolicyDocument = "{" +
    "\"Version\": \"2012-10-17\"," +
    "\"Statement\": [{" +
    "\"Effect\": \"Allow\"," +
    "\"Principal\": {" +
    "  \"AWS\": \"" + userArn + "\"" +
    "}," +
    "\"Action\": \"sts:AssumeRole\"" +
    "}]"}";

System.out.println(assumeRolePolicyDocument);
System.out.println(userName + " was successfully created.");
```

```
        System.out.println(DASHES);
        System.out.println("2. Creates a policy.");
        String polArn = createIAMPolicy(iam, policyName);
        System.out.println("The policy " + polArn + " was successfully created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Creates a role.");
        TimeUnit.SECONDS.sleep(30);
        String roleArn = createIAMRole(iam, roleName, assumeRolePolicyDocument);
        System.out.println(roleArn + " was successfully created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Grants the user permissions.");
        attachIAMRolePolicy(iam, roleName, polArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Wait for 30 secs so the resource is available");
        TimeUnit.SECONDS.sleep(30);
        System.out.println("5. Gets temporary credentials by assuming the role.");
        System.out.println("Perform an Amazon S3 Service operation using the
temporary credentials.");
        assumeRole(roleArn, roleSessionName, bucketName, accessKey, secretKey);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6 Getting ready to delete the AWS resources");
        deleteKey(iam, userName, accessKey);
        deleteRole(iam, roleName, polArn);
        deleteIAMUser(iam, userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("This IAM Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static AccessKey createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                .userName(user)
                .build();
```



```
        CreateAccessKeyResponse response = iam.createAccessKey(request);
        return response.accessKey();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static User createIAMUser(IamClient iam, String username) {
    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();
        CreateUserRequest request = CreateUserRequest.builder()
            .userName(username)
            .build();

        // Wait until the user is created.
        CreateUserResponse response = iam.createUser(request);
        GetUserRequest userRequest = GetUserRequest.builder()
            .userName(response.user().userName())
            .build();

        WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
        waitUntilUserExists.matched().response().ifPresent(System.out::println);
        return response.user();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static String createIAMRole(IamClient iam, String rolename, String json)
{

    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(json)
```

```
        .description("Created using the AWS SDK for Java")
        .build();

    CreateRoleResponse response = iam.createRole(request);
    System.out.println("The ARN of the role is " + response.role().arn());
    return response.role().arn();

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

public static String createIAMPolicy(IamClient iam, String policyName) {
    try {
        // Create an IamWaiter object.
        IamWaiter iamWaiter = iam.waiter();
        CreatePolicyRequest request = CreatePolicyRequest.builder()
            .policyName(policyName)
            .policyDocument(PolicyDocument).build();

        CreatePolicyResponse response = iam.createPolicy(request);
        GetPolicyRequest polRequest = GetPolicyRequest.builder()
            .policyArn(response.policy().arn())
            .build();

        WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
        return response.policy().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
    try {
```

```
        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
        .roleName(roleName)
        .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();
        String polArn;
        for (AttachedPolicy policy : attachedPolicies) {
            polArn = policy.policyArn();
            if (polArn.compareTo(policyArn) == 0) {
                System.out.println(roleName + " policy is already attached to
this role.");
                return;
            }
        }

        AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
        .roleName(roleName)
        .policyArn(policyArn)
        .build();

        iam.attachRolePolicy(attachRequest);
        System.out.println("Successfully attached policy " + policyArn + " to
role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Invoke an Amazon S3 operation using the Assumed Role.
public static void assumeRole(String roleArn, String roleSessionName, String
bucketName, String keyVal,
        String keySecret) {

    // Use the creds of the new IAM user that was created in this code example.
    AwsBasicCredentials credentials = AwsBasicCredentials.create(keyVal,
keySecret);
    StsClient stsClient = StsClient.builder()
        .region(Region.US_EAST_1)
```

```
        .credentialsProvider(StaticCredentialsProvider.create(credentials))
        .build();

    try {
        AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
            .roleArn(roleArn)
            .roleSessionName(roleSessionName)
            .build();

        AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
        Credentials myCreds = roleResponse.credentials();
        String key = myCreds.accessKeyId();
        String secKey = myCreds.secretAccessKey();
        String secToken = myCreds.sessionToken();

        // List all objects in an Amazon S3 bucket using the temp creds
retrieved by
        // invoking assumeRole.
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .credentialsProvider(
                StaticCredentialsProvider.create(AwsSessionCredentials.create(key, secKey,
                    secToken)))
            .region(region)
            .build();

        System.out.println("Created a S3Client using temp credentials.");
        System.out.println("Listing objects in " + bucketName);
        ListObjectsRequest listObjects = ListObjectsRequest.builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.println("The name of the key is " + myValue.key());
            System.out.println("The owner is " + myValue.owner());
        }

    } catch (StsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}

public static void deleteRole(IamClient iam, String roleName, String polArn) {

    try {
        // First the policy needs to be detached.
        DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
            .policyArn(polArn)
            .roleName(roleName)
            .build();

        iam.detachRolePolicy(rolePolicyRequest);

        // Delete the policy.
        DeletePolicyRequest request = DeletePolicyRequest.builder()
            .policyArn(polArn)
            .build();

        iam.deletePolicy(request);
        System.out.println("*** Successfully deleted " + polArn);

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
            .roleName(roleName)
            .build();

        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteKey(IamClient iam, String username, String accessKey) {
    try {
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
            .accessKeyId(accessKey)
            .userName(username)
            .build();

        iam.deleteAccessKey(request);
    }
}
```

```
        System.out.println("Successfully deleted access key " + accessKey +
            " from user " + username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteIAMUser(IamClient iam, String userName) {
    try {
        DeleteUserRequest request = DeleteUserRequest.builder()
            .userName(userName)
            .build();

        iam.deleteUser(request);
        System.out.println("*** Successfully deleted " + userName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)

- [PutUserPolicy](#)

## Arbeiten mit der IAM-Policy-Builder-API

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Richtlinien mithilfe der objektorientierten API.
- Verwenden Sie die IAM-Policy-Builder-API mit dem IAM-Service.

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

In den Beispielen werden folgende Importe verwendet.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.policybuilder.iam.IamConditionOperator;
import software.amazon.awssdk.policybuilder.iam.IamEffect;
import software.amazon.awssdk.policybuilder.iam.IamPolicy;
import software.amazon.awssdk.policybuilder.iam.IamPolicyWriter;
import software.amazon.awssdk.policybuilder.iam.IamPrincipal;
import software.amazon.awssdk.policybuilder.iam.IamPrincipalType;
import software.amazon.awssdk.policybuilder.iam.IamResource;
import software.amazon.awssdk.policybuilder.iam.IamStatement;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyVersionResponse;
import software.amazon.awssdk.services.sts.StsClient;

import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.List;
```

Erstellen Sie eine zeitbasierte Richtlinie.

```
public String timeBasedPolicyExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:GetItem")
            .addResource(IamResource.ALL)
            .addCondition(b1 -> b1

        .operator(IamConditionOperator.DATE_GREATER_THAN)

        .key("aws:CurrentTime")

        .value("2020-04-01T00:00:00Z"))

        .addCondition(b1 -> b1

        .operator(IamConditionOperator.DATE_LESS_THAN)

        .key("aws:CurrentTime")

        .value("2020-06-30T23:59:59Z")))
        .build();

    // Use an IamPolicyWriter to write out the JSON string to a more
readable
    // format.
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true)
        .build());
}
```

Erstellen Sie eine Richtlinie mit mehreren Bedingungen.

```
public String multipleConditionsExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:GetItem")
            .addAction("dynamodb:BatchGetItem")
```



```

        .addAction("dynamodb:Query")
        .addAction("dynamodb:PutItem")
        .addAction("dynamodb:UpdateItem")
        .addAction("dynamodb>DeleteItem")

        .addAction("dynamodb:BatchWriteItem")

        .addResource("arn:aws:dynamodb:*:*:table/table-name")

        .addConditions(IamConditionOperator.STRING_EQUALS

        .addPrefix("ForAllValues:"),

        "dynamodb:Attributes",

        List.of("column-
name1", "column-name2", "column-name3"))

        .addCondition(b1 -> b1

        .operator(IamConditionOperator.STRING_EQUALS

        .addSuffix("IfExists"))

        .key("dynamodb>Select")

        .value("SPECIFIC_ATTRIBUTES"))))

        .build();

        return policy.toJson(IamPolicyWriter.builder()

        .prettyPrint(true).build());

    }

```

Verwenden Sie Prinzipale in einer Richtlinie.

```

public String specifyPrincipalsExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.DENY)
            .addAction("s3:*")
            .addPrincipal(IamPrincipal.ALL)

        .addResource("arn:aws:s3::BUCKETNAME/*")

```

```

.addResource("arn:aws:s3:::BUCKETNAME")
                                .addCondition(b1 -> b1

.operator(IamConditionOperator.ARN_NOT_EQUALS)

.key("aws:PrincipalArn")

.value("arn:aws:iam::444455556666:user/user-name")))
                                .build();
return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
}

```

Gewähren Sie kontoübergreifenden -Zugriff.

```

public String allowCrossAccountAccessExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addPrincipal(IamPrincipalType.AWS,
"111122223333")
            .addAction("s3:PutObject")
            .addResource("arn:aws:s3:::DOC-
EXAMPLE-BUCKET/*")
            .addCondition(b1 -> b1

.operator(IamConditionOperator.STRING_EQUALS)
                                .key("s3:x-amz-acl")
                                .value("bucket-
owner-full-control")))
        .build();
return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}

```

Erstellen und laden Sie eine IamPolicy hoch.

```

public String createAndUploadPolicyExample(IamClient iam, String accountID,
String policyName) {
    // Build the policy.

```

```

        IamPolicy policy = IamPolicy.builder() // 'version' defaults to
"2012-10-17".
                .addStatement(IamStatement.builder()
                        .effect(IamEffect.ALLOW)
                        .addAction("dynamodb:PutItem")
                        .addResource("arn:aws:dynamodb:us-
east-1:" + accountID
                                + ":table/
exampleTableName")
                        .build())
                .build();
        // Upload the policy.
        iam.createPolicy(r ->
r.policyName(policyName).policyDocument(policy.toJson()));
        return
policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
    }

```

Downloaden und arbeiten Sie mit einer `IamPolicy`.

```

    public String createNewBasedOnExistingPolicyExample(IamClient iam, String
accountID, String policyName,
                String newPolicyName) {

        String policyArn = "arn:aws:iam::" + accountID + ":policy/" +
policyName;
        GetPolicyResponse getPolicyResponse = iam.getPolicy(r ->
r.policyArn(policyArn));

        String policyVersion =
getPolicyResponse.policy().defaultVersionId();
        GetPolicyVersionResponse getPolicyVersionResponse = iam
                .getPolicyVersion(r ->
r.policyArn(policyArn).versionId(policyVersion));

        // Create an IamPolicy instance from the JSON string returned from
IAM.
        String decodedPolicy =
URLDecoder.decode(getPolicyVersionResponse.policyVersion().document(),
                StandardCharsets.UTF_8);
        IamPolicy policy = IamPolicy.fromJson(decodedPolicy);
    }

```

```

        /*
         * All IamPolicy components are immutable, so use the copy method
that creates a
         * new instance that
         * can be altered in the same method call.
         *
         * Add the ability to get an item from DynamoDB as an additional
action.
        */
        IamStatement newStatement = policy.statements().get(0).copy(s ->
s.addAction("dynamodb:GetItem"));

        // Create a new statement that replaces the original statement.
        IamPolicy newPolicy = policy.copy(p ->
p.statements(Arrays.asList(newStatement)));

        // Upload the new policy. IAM now has both policies.
        iam.createPolicy(r -> r.policyName(newPolicyName)
            .policyDocument(newPolicy.toJson()));

        return
newPolicy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
    }

```

- Weitere Informationen finden Sie im [AWS SDK for Java 2.x -Entwicklerhandbuch](#).
- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [CreatePolicy](#)
  - [GetPolicy](#)
  - [GetPolicyVersion](#)

## AWS IoT -Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit verwenden AWS IoT.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

## Erste Schritte

### Hallo AWS IoT

Die folgenden Codebeispiele veranschaulichen, wie Sie mit der Verwendung von AWS IoT beginnen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.ListThingsRequest;
import software.amazon.awssdk.services.iot.model.ListThingsResponse;
import software.amazon.awssdk.services.iot.model.ThingAttribute;
import java.util.List;

public class HelloIoT {
    public static void main(String[] args) {
        System.out.println("Hello AWS IoT. Here is a listing of your AWS IoT
Things:");
        IotClient iotClient = IotClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllThings(iotClient);
    }

    public static void listAllThings( IotClient iotClient) {
        ListThingsRequest thingsRequest = ListThingsRequest.builder()
            .maxResults(10)
            .build();
```

```
ListThingsResponse response = iotClient.listThings(thingsRequest) ;
List<ThingAttribute> thingList = response.things();
for (ThingAttribute attribute : thingList) {
    System.out.println("Thing name: "+attribute.thingName());
    System.out.println("Thing ARN: "+attribute.thingArn());
}
}
```

- Weitere API-Informationen finden Sie unter [listThings](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Themen

- [Aktionen](#)
- [Szenarien](#)

## Aktionen

### Anfügen eines Zertifikats

Das folgende Codebeispiel zeigt, wie Sie ein - AWS IoT Zertifikat anfügen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void attachCertificateToThing(IotClient iotClient, String
thingName, String certificateArn) {
    // Attach the certificate to the thing.
    AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
    .thingName(thingName)
    .principal(certificateArn)
    .build();
```

```
AttachThingPrincipalResponse attachResponse =
iotClient.attachThingPrincipal(principalRequest);

// Verify the attachment was successful.
if (attachResponse.sdkHttpResponse().isSuccessful()) {
    System.out.println("Certificate attached to Thing successfully.");

    // Print additional information about the Thing.
    describeThing(iotClient, thingName);
} else {
    System.err.println("Failed to attach certificate to Thing. HTTP Status
Code: " +
        attachResponse.sdkHttpResponse().statusCode());
}
}
```

- Weitere API-Informationen finden Sie unter [AttachThingPrincipal](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erstellen eines Zertifikats

Das folgende Codebeispiel zeigt, wie Sie ein - AWS IoT Zertifikat erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createCertificate(IotClient iotClient) {
    try {
        CreateKeysAndCertificateResponse response =
iotClient.createKeysAndCertificate();
        String certificatePem = response.certificatePem();
        String certificateArn = response.certificateArn();

        // Print the details.
        System.out.println("\nCertificate:");
        System.out.println(certificatePem);
    }
}
```

```

        System.out.println("\nCertificate ARN:");
        System.out.println(certificateArn);
        return certificateArn;

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}

```

- Weitere API-Informationen finden Sie unter [CreateKeysAndCertificate](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erstellen einer Regel

Das folgende Codebeispiel zeigt, wie Sie eine - AWS IoT Regel erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

public static void createIoTRule(IotClient iotClient, String roleARN, String
ruleName, String action) {
    try {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();
    }
}

```



```
// Create the topic rule payload.
TopicRulePayload topicRulePayload = TopicRulePayload.builder()
    .sql(sql)
    .actions(myAction)
    .build();

// Create the topic rule request.
CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
    .ruleName(ruleName)
    .topicRulePayload(topicRulePayload)
    .build();

// Create the rule.
iotClient.createTopicRule(topicRuleRequest);
System.out.println("IoT Rule created successfully.");

} catch (IotException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [CreateTopicRule](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Ein Objekt erstellen

Das folgende Codebeispiel zeigt, wie Sie ein - AWS IoT Objekt erstellen.

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void createIoTThing(IotClient iotClient, String thingName) {
    try {
```

```
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
            .build();

        CreateThingResponse createThingResponse =
iotClient.createThing(createThingRequest);
        System.out.println(thingName + " was successfully created. The ARN value
is " + createThingResponse.thingArn());

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [CreateThing](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen eines Zertifikats

Das folgende Codebeispiel zeigt, wie Sie ein - AWS IoT Zertifikat löschen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void deleteCertificate(IotClient iotClient, String
certificateArn ) {
    DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
        .certificateId(extractCertificateId(certificateArn))
        .build();

    iotClient.deleteCertificate(certificateProviderRequest);
    System.out.println(certificateArn + " was successfully deleted.");
}
```

- Weitere API-Informationen finden Sie unter [DeleteCertificate](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Ein Objekt löschen

Das folgende Codebeispiel zeigt, wie Sie ein - AWS IoT Objekt löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void deleteIoTThing(IotClient iotClient, String thingName) {
    try {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        iotClient.deleteThing(deleteThingRequest);
        System.out.println("Deleted Thing " + thingName);


    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteThing](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beschreiben eines Objekts

Das folgende Codebeispiel zeigt, wie Sie ein - AWS IoT Objekt beschreiben.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
private static void describeThing(IotClient iotClient, String thingName) {
    try {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build() ;

        // Print Thing details.
        DescribeThingResponse describeResponse =
iotClient.describeThing(thingRequest);
        System.out.println("Thing Details:");
        System.out.println("Thing Name: " + describeResponse.thingName());
        System.out.println("Thing ARN: " + describeResponse.thingArn());

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeThing](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Trennen eines Zertifikats

Das folgende Codebeispiel zeigt, wie Sie ein - AWS IoT Zertifikat trennen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void detachThingPrincipal(IotClient iotClient, String thingName,
String certificateArn){
    try {
        DetachThingPrincipalRequest thingPrincipalRequest =
        DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();

        iotClient.detachThingPrincipal(thingPrincipalRequest);
        System.out.println(certificateArn +" was successfully removed from "
+thingName);

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DetachThingPrincipal](#) in der APIAWS SDK for Java 2.x -Referenz für .

### Abrufen von Endpunktinformationen

Das folgende Codebeispiel zeigt, wie AWS IoT Endpunktinformationen abgerufen werden.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String describeEndpoint(IotClient iotClient) {
    try {
        DescribeEndpointResponse endpointResponse =
            iotClient.describeEndpoint(DescribeEndpointRequest.builder().build());

        // Get the endpoint URL.
        String endpointUrl = endpointResponse.endpointAddress();
        String exString = getValue(endpointUrl);
        String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

        System.out.println("Full Endpoint URL: " + fullEndpoint);
        return fullEndpoint;

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Weitere API-Informationen finden Sie unter [DescribeEndpoint](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Auflisten Ihrer Zertifikate

Das folgende Codebeispiel zeigt, wie Sie Ihre AWS IoT Zertifikate auflisten.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.


```
public static void listCertificates(IotClient iotClient) {
    ListCertificatesResponse response = iotClient.listCertificates();
    List<Certificate> certList = response.certificates();
    for (Certificate cert : certList) {
        System.out.println("Cert id: " + cert.certificateId());
        System.out.println("Cert Arn: " + cert.certificateArn());
    }
}
```

- Weitere API-Informationen finden Sie unter [ListCertificates](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Abfragen des Suchindex

Das folgende Codebeispiel zeigt, wie Sie den AWS IoT Suchindex abfragen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void searchThings(IotClient iotClient, String queryString){
    SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
        .queryString(queryString)
        .build();

    try {
        // Perform the search and get the result.
    }
}
```

```
SearchIndexResponse searchIndexResponse =
iotClient.searchIndex(searchIndexRequest);

// Process the result.
if (searchIndexResponse.things().isEmpty()) {
    System.out.println("No things found.");
} else {
    searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
}
} catch (IotException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [SearchIndex](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Ein Objekt aktualisieren

Das folgende Codebeispiel zeigt, wie Sie ein - AWS IoT Objekt aktualisieren.

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void updateThing(IotClient iotClient, String thingName) {
    // Specify the new attribute values.
    String newLocation = "Office";
    String newFirmwareVersion = "v2.0";

    Map<String, String> attMap = new HashMap<>();
    attMap.put("location", newLocation);
    attMap.put("firmwareVersion", newFirmwareVersion);

    AttributePayload attributePayload = AttributePayload.builder()
```



```
        .attributes(attMap)
        .build();

UpdateThingRequest updateThingRequest = UpdateThingRequest.builder()
    .thingName(thingName)
    .attributePayload(attributePayload)
    .build();

try {
    // Update the IoT Thing attributes.
    iotClient.updateThing(updateThingRequest);
    System.out.println("Thing attributes updated successfully.");

} catch (IotException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [UpdateThing](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Szenarien

### Arbeiten mit Anwendungsfällen für die Geräteverwaltung

Das folgende Codebeispiel zeigt, wie Sie mit Anwendungsfällen für die AWS IoT Geräteverwaltung unter Verwendung des AWS IoT SDK arbeiten

#### SDK für Java 2.x

##### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
```

```
import software.amazon.awssdk.services.iot.model.Action;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.AttributePayload;
import software.amazon.awssdk.services.iot.model.Certificate;
import software.amazon.awssdk.services.iot.model.CreateKeysAndCertificateResponse;
import software.amazon.awssdk.services.iot.model.CreateThingRequest;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleRequest;
import software.amazon.awssdk.services.iot.model.DeleteCertificateRequest;
import software.amazon.awssdk.services.iot.model.CreateThingResponse;
import software.amazon.awssdk.services.iot.model.DeleteThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointResponse;
import software.amazon.awssdk.services.iot.model.DescribeThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeThingResponse;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.IotException;
import software.amazon.awssdk.services.iot.model.ListCertificatesResponse;
import software.amazon.awssdk.services.iot.model.ListTopicRulesRequest;
import software.amazon.awssdk.services.iot.model.ListTopicRulesResponse;
import software.amazon.awssdk.services.iot.model.SearchIndexRequest;
import software.amazon.awssdk.services.iot.model.SearchIndexResponse;
import software.amazon.awssdk.services.iot.model.SnsAction;
import software.amazon.awssdk.services.iot.model.TopicRuleListItem;
import software.amazon.awssdk.services.iot.model.TopicRulePayload;
import software.amazon.awssdk.services.iot.model.UpdateThingRequest;
import software.amazon.awssdk.services.iotdataplane.IotDataPlaneClient;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowRequest;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowResponse;
import software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowRequest;
import java.net.URI;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* This Java example performs these tasks:
*
* 1. Creates an AWS IoT Thing.
* 2. Generate and attach a device certificate.
* 3. Update an AWS IoT Thing with Attributes.
* 4. Get an AWS IoT Endpoint.
* 5. List your certificates.
* 6. Updates the shadow for the specified thing..
* 7. Write out the state information, in JSON format
* 8. Creates a rule
* 9. List rules
* 10. Search things
* 11. Detach and delete the certificate.
* 12. Delete Thing.
*/
public class IotScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String TOPIC = "your-iot-topic";
    public static void main(String[] args) {
        final String usage =
            """
            Usage:
                <roleARN> <snsAction>

            Where:
                roleARN - The ARN of an IAM role that has permission to work
with AWS IOT.
                snsAction - An ARN of an SNS topic.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String thingName;
        String ruleName;
        String roleARN = args[0];
        String snsAction = args[1];
        Scanner scanner = new Scanner(System.in);
        IotClient iotClient = IotClient.builder()
```

```
.region(Region.US_EAST_1)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the AWS IoT example workflow.");
System.out.println("""
    This example program demonstrates various interactions with the AWS
    Internet of Things (IoT) Core service. The program guides you through a series of
    steps,
        including creating an IoT Thing, generating a device certificate,
    updating the Thing with attributes, and so on.
    It utilizes the AWS SDK for Java V2 and incorporates functionality for
    creating and managing IoT Things, certificates, rules,
        shadows, and performing searches. The program aims to showcase AWS IoT
    capabilities and provides a comprehensive example for
        developers working with AWS IoT in a Java environment.

    """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create an AWS IoT Thing.");
System.out.println("""
    An AWS IoT Thing represents a virtual entity in the AWS IoT service that
    can be associated with a physical device.
    """);
// Prompt the user for input.
System.out.print("Enter Thing name: ");
thingName = scanner.nextLine();
createIoTThing(iotClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Generate a device certificate.");
System.out.println("""
    A device certificate performs a role in securing the communication
    between devices (Things) and the AWS IoT platform.
    """);

System.out.print("Do you want to create a certificate for " +thingName +"?
(y/n)");
String certAns = scanner.nextLine();
```

```
String certificateArn="" ;
if (certAns != null && certAns.trim().equalsIgnoreCase("y")) {
    certificateArn = createCertificate(iotClient);
    System.out.println("Attach the certificate to the AWS IoT Thing.");
    attachCertificateToThing(iotClient, thingName, certificateArn);
} else {
    System.out.println("A device certificate was not created.");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Update an AWS IoT Thing with Attributes.");
System.out.println("""
    IoT Thing attributes, represented as key-value pairs, offer a pivotal
advantage in facilitating efficient data
    management and retrieval within the AWS IoT ecosystem.
    """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
updateThing(iotClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Return a unique endpoint specific to the Amazon Web
Services account.");
System.out.println("""
    An IoT Endpoint refers to a specific URL or Uniform Resource Locator
that serves as the entry point for communication between IoT devices and the AWS
IoT service.
    """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
String endpointUrl = describeEndpoint(iotClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. List your AWS IoT certificates");
System.out.print("Press Enter to continue...");
scanner.nextLine();
if (certificateArn.length() > 0) {
    listCertificates(iotClient);
} else {
    System.out.println("You did not create a certificates. Skipping this
step.");
}
```

```
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Create an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
System.out.println("""
    A Thing Shadow refers to a feature that enables you to create a virtual
representation, or "shadow,"
    of a physical device or thing. The Thing Shadow allows you to
synchronize and control the state of a device between
    the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a Thing Shadow.
    """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
IotDataPlaneClient iotPlaneClient = IotDataPlaneClient.builder()
    .region(Region.US_EAST_1)
    .endpointOverride(URI.create(endpointUrl))
    .build();

updateShadowThing(iotPlaneClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Write out the state information, in JSON format.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
getPayload(iotPlaneClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Creates a rule");
System.out.println("""
Creates a rule that is an administrator-level action.
Any user who has permission to create rules will be able to access data
processed by the rule.
    """);
System.out.print("Enter Rule name: ");
ruleName = scanner.nextLine();
createIoTRule(iotClient, roleARN, ruleName, snsAction);
System.out.println(DASHES);

System.out.println(DASHES);
```

```
System.out.println("9. List your rules.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
listIoTRules(iotClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Search things using the Thing name.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
String queryString = "thingName:"+thingName ;
searchThings(iotClient, queryString);
System.out.println(DASHES);

System.out.println(DASHES);
if (certificateArn.length() > 0) {
    System.out.print("Do you want to detach and delete the certificate for "
+thingName +"? (y/n)");
    String delAns = scanner.nextLine();
    if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
        System.out.println("11. You selected to detach amd delete the
certificate.");
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        detachThingPrincipal(iotClient, thingName, certificateArn);
        deleteCertificate(iotClient, certificateArn);
    } else {
        System.out.println("11. You selected not to delete the
certificate.");
    }
} else {
    System.out.println("11. You did not create a certificate so there is
nothing to delete.");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Delete the AWS IoT Thing.");
System.out.print("Do you want to delete the IoT Thing? (y/n)");
String delAns = scanner.nextLine();
if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
    deleteIoTThing(iotClient, thingName);
} else {
    System.out.println("The IoT Thing was not deleted.");
}
```

```
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("The AWS IoT workflow has successfully completed.");
    System.out.println(DASHES);
}

public static void listCertificates(IotClient iotClient) {
    ListCertificatesResponse response = iotClient.listCertificates();
    List<Certificate> certList = response.certificates();
    for (Certificate cert : certList) {
        System.out.println("Cert id: " + cert.certificateId());
        System.out.println("Cert Arn: " + cert.certificateArn());
    }
}

public static void listIoTRules(IotClient iotClient) {
    try {
        ListTopicRulesRequest listTopicRulesRequest =
ListTopicRulesRequest.builder().build();
        ListTopicRulesResponse listTopicRulesResponse =
iotClient.listTopicRules(listTopicRulesRequest);
        System.out.println("List of IoT Rules:");
        List<TopicRuleListItem> ruleList = listTopicRulesResponse.rules();
        for (TopicRuleListItem rule : ruleList) {
            System.out.println("Rule Name: " + rule.ruleName());
            System.out.println("Rule ARN: " + rule.ruleArn());
            System.out.println("-----");
        }
    }

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createIoTRule(IotClient iotClient, String roleARN, String
ruleName, String action) {
    try {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
```



```
        .build();

    // Create the action.
    Action myAction = Action.builder()
        .sns(action1)
        .build();

    // Create the topic rule payload.
    TopicRulePayload topicRulePayload = TopicRulePayload.builder()
        .sql(sql)
        .actions(myAction)
        .build();

    // Create the topic rule request.
    CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
        .ruleName(ruleName)
        .topicRulePayload(topicRulePayload)
        .build();

    // Create the rule.
    iotClient.createTopicRule(topicRuleRequest);
    System.out.println("IoT Rule created successfully.");

    } catch (IotException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getPayload(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();

        GetThingShadowResponse getThingShadowResponse =
iotPlaneClient.getThingShadow(getThingShadowRequest);

        // Extracting payload from response.
        SdkBytes payload = getThingShadowResponse.payload();
        String payloadString = payload.asUtf8String();
```

```
        System.out.println("Received Shadow Data: " + payloadString);

    } catch (IotException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateShadowThing(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
        \"humidity\":50}}}\"";
        SdkBytes data= SdkBytes.fromString(stateDocument,
StandardCharsets.UTF_8 );
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
            .thingName(thingName)
            .payload(data)
            .build();

        // Update Thing Shadow.
        iotPlaneClient.updateThingShadow(updateThingShadowRequest);
        System.out.println("Thing Shadow updated successfully.");

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void updateThing(IotClient iotClient, String thingName) {
    // Specify the new attribute values.
    String newLocation = "Office";
    String newFirmwareVersion = "v2.0";

    Map<String, String> attMap = new HashMap<>();
    attMap.put("location", newLocation);
    attMap.put("firmwareVersion", newFirmwareVersion);

    AttributePayload attributePayload = AttributePayload.builder()
        .attributes(attMap)
        .build();
}
```

```
UpdateThingRequest updateThingRequest = UpdateThingRequest.builder()
    .thingName(thingName)
    .attributePayload(attributePayload)
    .build();

try {
    // Update the IoT Thing attributes.
    iotClient.updateThing(updateThingRequest);
    System.out.println("Thing attributes updated successfully.");

} catch (IotException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

}

public static String describeEndpoint(IotClient iotClient) {
    try {
        DescribeEndpointResponse endpointResponse =
iotClient.describeEndpoint(DescribeEndpointRequest.builder().build());

        // Get the endpoint URL.
        String endpointUrl = endpointResponse.endpointAddress();
        String exString = getValue(endpointUrl);
        String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

        System.out.println("Full Endpoint URL: " + fullEndpoint);
        return fullEndpoint;

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "" ;
}

public static void detachThingPrincipal(IotClient iotClient, String thingName,
String certificateArn){
    try {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
```

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

        iotClient.detachThingPrincipal(thingPrincipalRequest);
        System.out.println(certificateArn + " was successfully removed from "
+thingName);

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteCertificate(IotClient iotClient, String
certificateArn ) {
    DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
        .certificateId(extractCertificateId(certificateArn))
        .build();

    iotClient.deleteCertificate(certificateProviderRequest);
    System.out.println(certificateArn + " was successfully deleted.");
}

// Get the cert Id from the Cert ARN value.
private static String extractCertificateId(String certificateArn) {
    // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
    String[] arnParts = certificateArn.split(":");
    String certificateIdPart = arnParts[arnParts.length - 1];
    return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1);
}

public static String createCertificate(IotClient iotClient) {
    try {
        CreateKeysAndCertificateResponse response =
iotClient.createKeysAndCertificate();
        String certificatePem = response.certificatePem();
        String certificateArn = response.certificateArn();

        // Print the details.
        System.out.println("\nCertificate:");
        System.out.println(certificatePem);
        System.out.println("\nCertificate ARN:");
        System.out.println(certificateArn);
    }
}
```

```
        return certificateArn;

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}

public static void attachCertificateToThing(IotClient iotClient, String
thingName, String certificateArn) {
    // Attach the certificate to the thing.
    AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
        .thingName(thingName)
        .principal(certificateArn)
        .build();

    AttachThingPrincipalResponse attachResponse =
iotClient.attachThingPrincipal(principalRequest);

    // Verify the attachment was successful.
    if (attachResponse.sdkHttpResponse().isSuccessful()) {
        System.out.println("Certificate attached to Thing successfully.");

        // Print additional information about the Thing.
        describeThing(iotClient, thingName);
    } else {
        System.err.println("Failed to attach certificate to Thing. HTTP Status
Code: " +
            attachResponse.sdkHttpResponse().statusCode());
    }
}

private static void describeThing(IotClient iotClient, String thingName) {
    try {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        // Print Thing details.
        DescribeThingResponse describeResponse =
iotClient.describeThing(thingRequest);
```

```
        System.out.println("Thing Details:");
        System.out.println("Thing Name: " + describeResponse.thingName());
        System.out.println("Thing ARN: " + describeResponse.thingArn());

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteIoTThing(IotClient iotClient, String thingName) {
    try {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        iotClient.deleteThing(deleteThingRequest);
        System.out.println("Deleted Thing " + thingName);

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createIoTThing(IotClient iotClient, String thingName) {
    try {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
            .build();

        CreateThingResponse createThingResponse =
iotClient.createThing(createThingRequest);
        System.out.println(thingName + " was successfully created. The ARN value
is " + createThingResponse.thingArn());

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

private static String getValue(String input) {
    // Define a regular expression pattern for extracting the subdomain.
```

```
    Pattern pattern = Pattern.compile("^(.*)\\.\\.iot\\.\\.us-east-1\\.\\.amazonaws\\.\\.com");

    // Match the pattern against the input string.
    Matcher matcher = pattern.matcher(input);

    // Check if a match is found.
    if (matcher.find()) {
        // Extract the subdomain from the first capturing group.
        String subdomain = matcher.group(1);
        System.out.println("Extracted subdomain: " + subdomain);
        return subdomain ;
    } else {
        System.out.println("No match found");
    }
    return "" ;
}

public static void searchThings(IotClient iotClient, String queryString){
    SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
        .queryString(queryString)
        .build();

    try {
        // Perform the search and get the result.
        SearchIndexResponse searchIndexResponse =
        iotClient.searchIndex(searchIndexRequest);

        // Process the result.
        if (searchIndexResponse.things().isEmpty()) {
            System.out.println("No things found.");
        } else {
            searchIndexResponse.things().forEach(thing ->
            System.out.println("Thing id found using search is " + thing.thingId()));
        }
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

## AWS IoT data -Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit verwenden AWS IoT data.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)

### Aktionen

#### Abrufen des Schattens

Das folgende Codebeispiel zeigt, wie Sie den Schatten für ein - AWS IoT Objekt abrufen.

SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void getPayload(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        GetThingShadowRequest getThingShadowRequest =
        GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();
```



```
    GetThingShadowResponse getThingShadowResponse =
    iotPlaneClient.getThingShadow(getThingShadowRequest);

    // Extracting payload from response.
    SdkBytes payload = getThingShadowResponse.payload();
    String payloadString = payload.asUtf8String();
    System.out.println("Received Shadow Data: " + payloadString);

} catch (IotException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [GetThingShadow](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Aktualisieren des Schattens

Das folgende Codebeispiel zeigt, wie Sie den Schatten für ein - AWS IoT Objekt aktualisieren.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void updateShadowThing(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
        \"humidity\":50}}}\"";
        SdkBytes data= SdkBytes.fromString(stateDocument,
        StandardCharsets.UTF_8 );
        UpdateThingShadowRequest updateThingShadowRequest =
        UpdateThingShadowRequest.builder()
```

```
        .thingName(thingName)
        .payload(data)
        .build();

    // Update Thing Shadow.
    iotPlaneClient.updateThingShadow(updateThingShadowRequest);
    System.out.println("Thing Shadow updated successfully.");

} catch (IotException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [UpdateThingShadow](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Beispiele für Amazon Keyspaces mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon Keyspaces verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Erste Schritte

#### Hallo Amazon Keyspaces

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon Keyspaces.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.keyspaces.KeyspacesClient;
import software.amazon.awssdk.services.keyspaces.model.KeyspaceSummary;
import software.amazon.awssdk.services.keyspaces.model.KeyspacesException;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesRequest;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesResponse;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloKeyspaces {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        KeyspacesClient keyClient = KeyspacesClient.builder()
            .region(region)
            .build();

        listKeyspaces(keyClient);
    }

    public static void listKeyspaces(KeyspacesClient keyClient) {
        try {
            ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
                .maxResults(10)
                .build();

            ListKeyspacesResponse response =
                keyClient.listKeyspaces(keyspacesRequest);
        }
    }
}
```

```
        List<KeyspaceSummary> keyspaces = response.keyspaces();
        for (KeyspaceSummary keyspace : keyspaces) {
            System.out.println("The name of the keyspace is " +
                keyspace.keyspaceName());
        }

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [ListKeyspaces](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Themen

- [Aktionen](#)
- [Szenarien](#)

## Aktionen

### Erstellen eines Keyspace

Das folgende Codebeispiel zeigt, wie Sie einen Amazon Keyspaces Keyspace erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
```

```
        .build();

        CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
        System.out.println("The ARN of the KeySpace is " +
response.resourceArn());

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [CreateKeyspace](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen einer Tabelle

Das folgende Codebeispiel zeigt, wie Sie eine Amazon Keyspaces-Tabelle erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        // Set the columns.
        ColumnDefinition defTitle = ColumnDefinition.builder()
            .name("title")
            .type("text")
            .build();

        ColumnDefinition defYear = ColumnDefinition.builder()
            .name("year")
            .type("int")
            .build();
```

```
ColumnDefinition defReleaseDate = ColumnDefinition.builder()
    .name("release_date")
    .type("timestamp")
    .build();

ColumnDefinition defPlot = ColumnDefinition.builder()
    .name("plot")
    .type("text")
    .build();

List<ColumnDefinition> collList = new ArrayList<>();
collList.add(defTitle);
collList.add(defYear);
collList.add(defReleaseDate);
collList.add(defPlot);

// Set the keys.
PartitionKey yearKey = PartitionKey.builder()
    .name("year")
    .build();

PartitionKey titleKey = PartitionKey.builder()
    .name("title")
    .build();

List<PartitionKey> keyList = new ArrayList<>();
keyList.add(yearKey);
keyList.add(titleKey);

SchemaDefinition schemaDefinition = SchemaDefinition.builder()
    .partitionKeys(keyList)
    .allColumns(collList)
    .build();

PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
    .status(PointInTimeRecoveryStatus.ENABLED)
    .build();

CreateTableRequest tableRequest = CreateTableRequest.builder()
    .keyspaceName(keySpace)
    .tableName(tableName)
    .schemaDefinition(schemaDefinition)
    .pointInTimeRecovery(timeRecovery)
```

```
        .build();

        CreateTableResponse response = keyClient.createTable(tableRequest);
        System.out.println("The table ARN is " + response.resourceArn());

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [CreateTable](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen eines Keyspace

Das folgende Codebeispiel zeigt, wie Sie einen Amazon Keyspaces-Keyspace löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        keyClient.deleteKeyspace(deleteKeyspaceRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteKeyspace](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen einer Tabelle

Das folgende Codebeispiel zeigt, wie Sie eine Amazon Keyspaces-Tabelle löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
String tableName) {
    try {
        DeleteTableRequest tableRequest = DeleteTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        keyClient.deleteTable(tableRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteTable](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Abrufen von Daten zu einem Keyspace

Das folgende Codebeispiel zeigt, wie Sie Daten zu einem Amazon Keyspaces-Keyspace abrufen.



## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);
        String name = response.keyspaceName();
        System.out.println("The " + name + " KeySpace is ready");


    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [GetKeyspace](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Abrufen von Daten zu einer Tabelle

Das folgende Codebeispiel zeigt, wie Sie Daten zu einer Amazon Keyspaces-Tabelle abrufen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
    throws InterruptedException {
    try {
        boolean tableStatus = false;
        String status;
        GetTableResponse response = null;
        GetTableRequest tableRequest = GetTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        while (!tableStatus) {
            response = keyClient.getTable(tableRequest);
            status = response.statusAsString();
            System.out.println(". The table status is " + status);

            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true;
            }
            Thread.sleep(500);
        }

        List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
        for (ColumnDefinition def : cols) {
            System.out.println("The column name is " + def.name());
            System.out.println("The column type is " + def.type());
        }

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}


```

- Weitere API-Informationen finden Sie unter [GetTable](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Keyspaces

Das folgende Codebeispiel zeigt, wie Sie Amazon Keyspaces Keyspaces auflisten.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
    try {
        ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
            .maxResults(10)
            .build();

        ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
        listRes.stream()
            .flatMap(r -> r.keyspaces().stream())
            .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [ListKeyspaces](#) in der APIAWS SDK for Java 2.x - Referenz für .

### Auflisten von Tabellen in einem Keyspace

Das folgende Codebeispiel zeigt, wie Sie Amazon-Keyspaces-Tabellen in einem Keyspace auflisten.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void listTables(KeyspacesClient keyClient, String keyspaceName) {
    try {
        ListTablesRequest tablesRequest = ListTablesRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
        listRes.stream()
            .flatMap(r -> r.tables().stream())
            .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
                " Table name: " + content.tableName()));

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [ListTables](#) in der APIAWS SDK for Java 2.x - Referenz für .

### Wiederherstellen einer Tabelle zu einem bestimmten Zeitpunkt

Das folgende Codebeispiel zeigt, wie eine Amazon Keyspaces-Tabelle zu einem bestimmten Zeitpunkt wiederhergestellt wird.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void restoreTable(KeyspacesClient keyClient, String keyspaceName,
    ZonedDateTime utc) {
    try {
        Instant myTime = utc.toInstant();
        RestoreTableRequest restoreTableRequest = RestoreTableRequest.builder()
            .restoreTimestamp(myTime)
            .sourceTableName("Movie")
            .targetKeyspaceName(keyspaceName)
            .targetTableName("MovieRestore")
            .sourceKeyspaceName(keyspaceName)
            .build();

        RestoreTableResponse response =
            keyClient.restoreTable(restoreTableRequest);
        System.out.println("The ARN of the restored table is " +
            response.restoredTableARN());

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [RestoreTable](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Aktualisieren einer Tabelle

Das folgende Codebeispiel zeigt, wie Sie eine Amazon Keyspaces-Tabelle aktualisieren.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        ColumnDefinition def = ColumnDefinition.builder()
            .name("watched")
            .type("boolean")
            .build();

        UpdateTableRequest tableRequest = UpdateTableRequest.builder()
            .keyspaceName(keySpace)
            .tableName(tableName)
            .addColumnns(def)
            .build();

        keyClient.updateTable(tableRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [UpdateTable](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Szenarien

### Erste Schritte mit Keyspaces und Tabellen

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 angesehene Filme zu verfolgen.
- Setzen Sie die Tabelle in den vorherigen Zustand zurück und bereinigen Sie Ressourcen.

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * Before running this Java code example, you must create a
 * Java keystore (JKS) file and place it in your project's resources folder.
 *
 * This file is a secure file format used to hold certificate information for
 * Java applications. This is required to make a connection to Amazon Keyspaces.
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/keyspaces/latest/devguide/using_java_driver.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Create a keyspace.
 * 2. Check for keyspace existence.
 * 3. List keyspaces using a paginator.
 * 4. Create a table with a simple movie data schema and enable point-in-time
```

```
* recovery.
* 5. Check for the table to be in an Active state.
* 6. List all tables in the keyspace.
* 7. Use a Cassandra driver to insert some records into the Movie table.
* 8. Get all records from the Movie table.
* 9. Get a specific Movie.
* 10. Get a UTC timestamp for the current time.
* 11. Update the table schema to add a 'watched' Boolean column.
* 12. Update an item as watched.
* 13. Query for items with watched = True.
* 14. Restore the table back to the previous state using the timestamp.
* 15. Check for completion of the restore action.
* 16. Delete the table.
* 17. Confirm that both tables are deleted.
* 18. Delete the keyspace.
*/

public class ScenarioKeyspaces {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    /*
    * Usage:
    * fileName - The name of the JSON file that contains movie data. (Get this file
    * from the GitHub repo at resources/sample_file.)
    * keyspaceName - The name of the keyspace to create.
    */
    public static void main(String[] args) throws InterruptedException, IOException
    {
        String fileName = "<Replace with the JSON file that contains movie data>";
        String keyspaceName = "<Replace with the name of the keyspace to create>";
        String titleUpdate = "The Family";
        int yearUpdate = 2013;
        String tableName = "Movie";
        String tableNameRestore = "MovieRestore";
        Region region = Region.US_EAST_1;
        KeyspacesClient keyClient = KeyspacesClient.builder()
            .region(region)
            .build();

        DriverConfigLoader loader =
DriverConfigLoader.fromClasspath("application.conf");
        CqlSession session = CqlSession.builder()
            .withConfigLoader(loader)
            .build();
```



```
System.out.println(DASHES);
System.out.println("Welcome to the Amazon Keyspaces example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create a keyspace.");
createKeyspace(keyClient, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
Thread.sleep(5000);
System.out.println("2. Check for keyspace existence.");
checkKeyspaceExistence(keyClient, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. List keyspaces using a paginator.");
listKeyspacesPaginator(keyClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Create a table with a simple movie data schema and
enable point-in-time recovery.");
createTable(keyClient, keyspaceName, tableName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Check for the table to be in an Active state.");
Thread.sleep(6000);
checkTable(keyClient, keyspaceName, tableName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. List all tables in the keyspace.");
listTables(keyClient, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Use a Cassandra driver to insert some records into
the Movie table.");
Thread.sleep(6000);
loadData(session, fileName, keyspaceName);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("8. Get all records from the Movie table.");
getMovieData(session, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Get a specific Movie.");
getSpecificMovie(session, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get a UTC timestamp for the current time.");
ZonedDateTime utc = ZonedDateTime.now(ZoneOffset.UTC);
System.out.println("DATETIME = " + Date.from(utc.toInstant()));
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Update the table schema to add a watched Boolean
column.");
updateTable(keyClient, keyspaceName, tableName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Update an item as watched.");
Thread.sleep(10000); // Wait 10 secs for the update.
updateRecord(session, keyspaceName, titleUpdate, yearUpdate);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Query for items with watched = True.");
getWatchedData(session, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Restore the table back to the previous state using
the timestamp.");
System.out.println("Note that the restore operation can take up to 20
minutes.");
restoreTable(keyClient, keyspaceName, utc);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("15. Check for completion of the restore action.");
```

```
Thread.sleep(5000);
checkRestoredTable(keyClient, keyspaceName, "MovieRestore");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("16. Delete both tables.");
deleteTable(keyClient, keyspaceName, tableName);
deleteTable(keyClient, keyspaceName, tableNameRestore);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("17. Confirm that both tables are deleted.");
checkTableDelete(keyClient, keyspaceName, tableName);
checkTableDelete(keyClient, keyspaceName, tableNameRestore);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("18. Delete the keyspace.");
deleteKeyspace(keyClient, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The scenario has completed successfully.");
System.out.println(DASHES);
}

public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        keyClient.deleteKeyspace(deleteKeyspaceRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void checkTableDelete(KeyspacesClient keyClient, String
keyspaceName, String tableName)
```

```
        throws InterruptedException {
    try {
        String status;
        GetTableResponse response;
        GetTableRequest tableRequest = GetTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        // Keep looping until table cannot be found and a
ResourceNotFoundException is
        // thrown.
        while (true) {
            response = keyClient.getTable(tableRequest);
            status = response.statusAsString();
            System.out.println(". The table status is " + status);
            Thread.sleep(500);
        }

    } catch (ResourceNotFoundException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println("The table is deleted");
}

public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
String tableName) {
    try {
        DeleteTableRequest tableRequest = DeleteTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        keyClient.deleteTable(tableRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void checkRestoredTable(KeyspacesClient keyClient, String
keyspaceName, String tableName)
    throws InterruptedException {
```

```
try {
    boolean tableStatus = false;
    String status;
    GetTableResponse response = null;
    GetTableRequest tableRequest = GetTableRequest.builder()
        .keyspaceName(keyspaceName)
        .tableName(tableName)
        .build();

    while (!tableStatus) {
        response = keyClient.getTable(tableRequest);
        status = response.statusAsString();
        System.out.println("The table status is " + status);

        if (status.compareTo("ACTIVE") == 0) {
            tableStatus = true;
        }
        Thread.sleep(500);
    }

    List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
    for (ColumnDefinition def : cols) {
        System.out.println("The column name is " + def.name());
        System.out.println("The column type is " + def.type());
    }

} catch (KeyspacesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

}

public static void restoreTable(KeyspacesClient keyClient, String keyspaceName,
    ZonedDateTime utc) {
    try {
        Instant myTime = utc.toInstant();
        RestoreTableRequest restoreTableRequest = RestoreTableRequest.builder()
            .restoreTimestamp(myTime)
            .sourceTableName("Movie")
            .targetKeyspaceName(keyspaceName)
            .targetTableName("MovieRestore")
            .sourceKeyspaceName(keyspaceName)
            .build();
```

```

        RestoreTableResponse response =
keyClient.restoreTable(restoreTableRequest);
        System.out.println("The ARN of the restored table is " +
response.restoredTableARN());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getWatchedData(CqlSession session, String keyspaceName) {
        ResultSet resultSet = session
            .execute("SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE
watched = true ALLOW FILTERING;");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    public static void updateRecord(CqlSession session, String keySpace, String
titleUpdate, int yearUpdate) {
        String sqlStatement = "UPDATE \"" + keySpace
            + "\".\"Movie\" SET watched=true WHERE title = :k0 AND year = :k1;";
        BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
        PreparedStatement preparedStatement = session.prepare(sqlStatement);
        builder.addStatement(preparedStatement.boundStatementBuilder()
            .setString("k0", titleUpdate)
            .setInt("k1", yearUpdate)
            .build());

        BatchStatement batchStatement = builder.build();
        session.execute(batchStatement);
    }

    public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
        try {
            ColumnDefinition def = ColumnDefinition.builder()
                .name("watched")

```

```
        .type("boolean")
        .build();

    UpdateTableRequest tableRequest = UpdateTableRequest.builder()
        .keyspaceName(keySpace)
        .tableName(tableName)
        .addColumnns(def)
        .build();

    keyClient.updateTable(tableRequest);

} catch (KeyspacesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

}

public static void getSpecificMovie(CqlSession session, String keyspaceName) {
    ResultSet resultSet = session.execute(
        "SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE title = 'The
Family' ALLOW FILTERING ;");
    resultSet.forEach(item -> {
        System.out.println("The Movie title is " + item.getString("title"));
        System.out.println("The Movie year is " + item.getInt("year"));
        System.out.println("The plot is " + item.getString("plot"));
    });
}

// Get records from the Movie table.
public static void getMovieData(CqlSession session, String keyspaceName) {
    ResultSet resultSet = session.execute("SELECT * FROM \"" + keyspaceName +
"\".\"Movie\";");
    resultSet.forEach(item -> {
        System.out.println("The Movie title is " + item.getString("title"));
        System.out.println("The Movie year is " + item.getInt("year"));
        System.out.println("The plot is " + item.getString("plot"));
    });
}

// Load data into the table.
public static void loadData(CqlSession session, String fileName, String
keySpace) throws IOException {
    String sqlStatement = "INSERT INTO \"" + keySpace + "\".\"Movie\" (title,
year, plot) values (:k0, :k1, :k2)";
```

```
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    while (iter.hasNext()) {

        // Add 20 movies to the table.
        if (t == 20)
            break;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String plot = currentNode.path("info").path("plot").toString();

        // Insert the data into the Amazon Keyspaces table.
        BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
        PreparedStatement preparedStatement = session.prepare(sqlStatement);
        builder.addStatement(preparedStatement.boundStatementBuilder()
            .setString("k0", title)
            .setInt("k1", year)
            .setString("k2", plot)
            .build());

        BatchStatement batchStatement = builder.build();
        session.execute(batchStatement);
        t++;
    }

    System.out.println("You have added " + t + " records successfully!");
}

public static void listTables(KeyspacesClient keyClient, String keyspaceName) {
    try {
        ListTablesRequest tablesRequest = ListTablesRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
    }
}
```



```
        listRes.stream()
            .flatMap(r -> r.tables().stream())
            .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
                " Table name: " + content.tableName()));

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
    throws InterruptedException {
    try {
        boolean tableStatus = false;
        String status;
        GetTableResponse response = null;
        GetTableRequest tableRequest = GetTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        while (!tableStatus) {
            response = keyClient.getTable(tableRequest);
            status = response.statusAsString();
            System.out.println(". The table status is " + status);

            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true;
            }
            Thread.sleep(500);
        }

        List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
        for (ColumnDefinition def : cols) {
            System.out.println("The column name is " + def.name());
            System.out.println("The column type is " + def.type());
        }

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
  }  
  
  public static void createTable(KeyspacesClient keyClient, String keySpace,  
String tableName) {  
    try {  
      // Set the columns.  
      ColumnDefinition defTitle = ColumnDefinition.builder()  
        .name("title")  
        .type("text")  
        .build();  
  
      ColumnDefinition defYear = ColumnDefinition.builder()  
        .name("year")  
        .type("int")  
        .build();  
  
      ColumnDefinition defReleaseDate = ColumnDefinition.builder()  
        .name("release_date")  
        .type("timestamp")  
        .build();  
  
      ColumnDefinition defPlot = ColumnDefinition.builder()  
        .name("plot")  
        .type("text")  
        .build();  
  
      List<ColumnDefinition> collList = new ArrayList<>();  
      collList.add(defTitle);  
      collList.add(defYear);  
      collList.add(defReleaseDate);  
      collList.add(defPlot);  
  
      // Set the keys.  
      PartitionKey yearKey = PartitionKey.builder()  
        .name("year")  
        .build();  
  
      PartitionKey titleKey = PartitionKey.builder()  
        .name("title")  
        .build();  
  
      List<PartitionKey> keyList = new ArrayList<>();  
      keyList.add(yearKey);
```

```
        keyList.add(titleKey);

        SchemaDefinition schemaDefinition = SchemaDefinition.builder()
            .partitionKeys(keyList)
            .allColumns(collList)
            .build();

        PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
            .status(PointInTimeRecoveryStatus.ENABLED)
            .build();

        CreateTableRequest tableRequest = CreateTableRequest.builder()
            .keyspaceName(keySpace)
            .tableName(tableName)
            .schemaDefinition(schemaDefinition)
            .pointInTimeRecovery(timeRecovery)
            .build();

        CreateTableResponse response = keyClient.createTable(tableRequest);
        System.out.println("The table ARN is " + response.resourceArn());

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
    try {
        ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
            .maxResults(10)
            .build();

        ListKeyspacesIterable listRes =
            keyClient.listKeyspacesPaginator(keyspacesRequest);
        listRes.stream()
            .flatMap(r -> r.keyspaces().stream())
            .forEach(content -> System.out.println(" Name: " +
                content.keyspaceName()));

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }

    public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
                .keyspaceName(keyspaceName)
                .build();

            GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);
            String name = response.keyspaceName();
            System.out.println("The " + name + " KeySpace is ready");

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
                .keyspaceName(keyspaceName)
                .build();

            CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
            System.out.println("The ARN of the KeySpace is " +
response.resourceArn());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [CreateKeyspace](#)
  - [CreateTable](#)

- [DeleteKeyspace](#)
- [DeleteTable](#)
- [GetKeyspace](#)
- [GetTable](#)
- [ListKeyspaces](#)
- [ListTables](#)
- [RestoreTable](#)
- [UpdateTable](#)

## Kinesis-Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Kinesis verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Themen

- [Aktionen](#)
- [Serverless-Beispiele](#)

## Aktionen

### Erstellen eines -Streams

Das folgende Codebeispiel zeigt, wie Sie einen Kinesis-Stream erstellen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.CreateStreamRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateDataStream {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <streamName>

            Where:
                streamName - The Amazon Kinesis data stream (for example,
                StockTradeStream).
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
```

```
        .region(region)
        .build();
    createStream(kinesisClient, streamName);
    System.out.println("Done");
    kinesisClient.close();
}

public static void createStream(KinesisClient kinesisClient, String streamName)
{
    try {
        CreateStreamRequest streamReq = CreateStreamRequest.builder()
            .streamName(streamName)
            .shardCount(1)
            .build();

        kinesisClient.createStream(streamReq);

    } catch (KinesisException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [CreateStream](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen eines Streams

Das folgende Codebeispiel zeigt, wie Sie einen Kinesis-Stream löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
```

```
import software.amazon.awssdk.services.kinesis.model.DeleteStreamRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDataStream {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <streamName>

            Where:
                streamName - The Amazon Kinesis data stream (for example,
StockTradeStream)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
            .region(region)
            .build();

        deleteStream(kinesisClient, streamName);
        kinesisClient.close();
        System.out.println("Done");
    }

    public static void deleteStream(KinesisClient kinesisClient, String streamName)
    {
        try {
            DeleteStreamRequest delStream = DeleteStreamRequest.builder()
```



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

    kinesisClient.deleteStream(delStream);

} catch (KinesisException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [DeleteStream](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Abrufen von Daten in Batches aus einem Stream

Das folgende Codebeispiel zeigt, wie Daten in Batches aus einem Kinesis-Stream abgerufen werden.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.Shard;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorRequest;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorResponse;
import software.amazon.awssdk.services.kinesis.model.Record;
import software.amazon.awssdk.services.kinesis.model.GetRecordsRequest;
import software.amazon.awssdk.services.kinesis.model.GetRecordsResponse;
import java.util.ArrayList;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetRecords {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <streamName>

                Where:
                streamName - The Amazon Kinesis data stream to read from (for
example, StockTradeStream).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
            .region(region)
            .build();

        getStockTrades(kinesisClient, streamName);
        kinesisClient.close();
    }

    public static void getStockTrades(KinesisClient kinesisClient, String
streamName) {
        String shardIterator;
        String lastShardId = null;
        DescribeStreamRequest describeStreamRequest =
DescribeStreamRequest.builder()
            .streamName(streamName)
            .build();
    }
}
```

```
List<Shard> shards = new ArrayList<>();
DescribeStreamResponse streamRes;
do {
    streamRes = kinesisClient.describeStream(describeStreamRequest);
    shards.addAll(streamRes.streamDescription().shards());

    if (shards.size() > 0) {
        lastShardId = shards.get(shards.size() - 1).shardId();
    }
} while (streamRes.streamDescription().hasMoreShards());

GetShardIteratorRequest itReq = GetShardIteratorRequest.builder()
    .streamName(streamName)
    .shardIteratorType("TRIM_HORIZON")
    .shardId(lastShardId)
    .build();

GetShardIteratorResponse shardIteratorResult =
kinesisClient.getShardIterator(itReq);
shardIterator = shardIteratorResult.shardIterator();

// Continuously read data records from shard.
List<Record> records;

// Create new GetRecordsRequest with existing shardIterator.
// Set maximum records to return to 1000.
GetRecordsRequest recordsRequest = GetRecordsRequest.builder()
    .shardIterator(shardIterator)
    .limit(1000)
    .build();

GetRecordsResponse result = kinesisClient.getRecords(recordsRequest);

// Put result into record list. Result may be empty.
records = result.records();

// Print records
for (Record record : records) {
    SdkBytes byteBuffer = record.data();
    System.out.printf("Seq No: %s - %s%n", record.sequenceNumber(), new
String(byteBuffer.asByteArray()));
}
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [GetRecords](#)
  - [GetShardIterator](#)

## Einfügen von Daten in einen Stream

Das folgende Codebeispiel zeigt, wie Daten in einen Kinesis-Stream eingefügt werden.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StockTradesWriter {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <streamName>
```

```
        Where:
            streamName - The Amazon Kinesis data stream to which records are
written (for example, StockTradeStream)
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String streamName = args[0];
    Region region = Region.US_EAST_1;
    KinesisClient kinesisClient = KinesisClient.builder()
        .region(region)
        .build();

    // Ensure that the Kinesis Stream is valid.
    validateStream(kinesisClient, streamName);
    setStockData(kinesisClient, streamName);
    kinesisClient.close();
}

public static void setStockData(KinesisClient kinesisClient, String streamName)
{
    try {
        // Repeatedly send stock trades with a 100 milliseconds wait in between.
        StockTradeGenerator stockTradeGenerator = new StockTradeGenerator();

        // Put in 50 Records for this example.
        int index = 50;
        for (int x = 0; x < index; x++) {
            StockTrade trade = stockTradeGenerator.getRandomTrade();
            sendStockTrade(trade, kinesisClient, streamName);
            Thread.sleep(100);
        }

    } catch (KinesisException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
```

```

    private static void sendStockTrade(StockTrade trade, KinesisClient
kinesisClient,
        String streamName) {
        byte[] bytes = trade.toJsonAsBytes();

        // The bytes could be null if there is an issue with the JSON serialization
by
        // the Jackson JSON library.
        if (bytes == null) {
            System.out.println("Could not get JSON bytes for stock trade");
            return;
        }

        System.out.println("Putting trade: " + trade);
        PutRecordRequest request = PutRecordRequest.builder()
            .partitionKey(trade.getTickerSymbol()) // We use the ticker symbol
as the partition key, explained in
                                                    // the Supplemental
Information section below.
            .streamName(streamName)
            .data(SdkBytes.fromByteArray(bytes))
            .build();

        try {
            kinesisClient.putRecord(request);
        } catch (KinesisException e) {
            System.err.println(e.getMessage());
        }
    }

    private static void validateStream(KinesisClient kinesisClient, String
streamName) {
        try {
            DescribeStreamRequest describeStreamRequest =
DescribeStreamRequest.builder()
                .streamName(streamName)
                .build();

            DescribeStreamResponse describeStreamResponse =
kinesisClient.describeStream(describeStreamRequest);

            if (!
describeStreamResponse.streamDescription().streamStatus().toString().equals("ACTIVE"))
        {

```

```
        System.err.println("Stream " + streamName + " is not active. Please  
wait a few moments and try again.");  
        System.exit(1);  
    }  
  
    } catch (KinesisException e) {  
        System.err.println("Error found while describing the stream " +  
streamName);  
        System.err.println(e);  
        System.exit(1);  
    }  
}  
}
```

- Weitere API-Informationen finden Sie unter [PutRecord](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Serverless-Beispiele

### Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöst wird. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

### Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Java.

```
package example;  
  
import com.amazonaws.services.lambda.runtime.Context;  
import com.amazonaws.services.lambda.runtime.LambdaLogger;  
import com.amazonaws.services.lambda.runtime.RequestHandler;
```

```
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;

public class Handler implements RequestHandler<KinesisEvent, Void> {
    @Override
    public Void handleRequest(final KinesisEvent event, final Context context) {
        LambdaLogger logger = context.getLogger();
        if (event.getRecords().isEmpty()) {
            logger.log("Empty Kinesis Event received");
            return null;
        }
        for (KinesisEvent.KinesisEventRecord record : event.getRecords()) {
            try {
                logger.log("Processed Event with EventId: "+record.getEventID());
                String data = new String(record.getKinesis().getData().array());
                logger.log("Data:"+ data);
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex) {
                logger.log("An error occurred:"+ex.getMessage());
                throw ex;
            }
        }
        logger.log("Successfully processed:"+event.getRecords().size()+" records");
        return null;
    }
}
```

## Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine partielle Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.



## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,
StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(KinesisEvent input, Context context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
ArrayList<>();
        String curRecordSequenceNumber = "";

        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :
input.getRecords()) {
            try {
                //Process your record
                KinesisEvent.Record kinesisRecord = kinesisEventRecord.getKinesis();
                curRecordSequenceNumber = kinesisRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
immediately.
                Lambda will immediately begin to retry processing from this
failed item onwards. */
                batchItemFailures.add(new
StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
```

```
        return new StreamsEventResponse(batchItemFailures);
    }
}

return new StreamsEventResponse(batchItemFailures);
}
}
```

## AWS KMS -Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit verwenden AWS KMS.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)

### Aktionen

Erstellen einer Erteilung für einen Schlüssel

Das folgende Codebeispiel zeigt, wie Sie eine Erteilung für einen KMS-Schlüssel erstellen.

SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.CreateGrantRequest;
import software.amazon.awssdk.services.kms.model.CreateGrantResponse;
import software.amazon.awssdk.services.kms.model.KmsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateGrant {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <keyId> <granteePrincipal> <operation>\s

            Where:
                keyId - The unique identifier for the customer master key (CMK)
that the grant applies to.\s
                granteePrincipal - The principal that is given permission to
perform the operations that the grant permits.\s
                operation - An operation (for example, Encrypt).\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String keyId = args[0];
        String granteePrincipal = args[1];
        String operation = args[2];
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        String grantId = createGrant(kmsClient, keyId, granteePrincipal, operation);
    }
}
```

```
        System.out.printf("Successfully created a grant with ID %s%n", grantId);
        kmsClient.close();
    }

    public static String createGrant(KmsClient kmsClient, String keyId, String
granteePrincipal, String operation) {
        try {
            CreateGrantRequest grantRequest = CreateGrantRequest.builder()
                .keyId(keyId)
                .granteePrincipal(granteePrincipal)
                .operationsWithStrings(operation)
                .build();

            CreateGrantResponse response = kmsClient.createGrant(grantRequest);
            return response.grantId();

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Weitere API-Informationen finden Sie unter [CreateGrant](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen eines -Schlüssels

Das folgende Codebeispiel zeigt, wie Sie ein erstellen AWS KMS key.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
```

```
import software.amazon.awssdk.services.kms.model.CreateKeyRequest;
import software.amazon.awssdk.services.kms.model.CustomerMasterKeySpec;
import software.amazon.awssdk.services.kms.model.CreateKeyResponse;
import software.amazon.awssdk.services.kms.model.KmsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCustomerKey {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        String keyDesc = "Created by the AWS KMS API";
        System.out.println("The key id is " + createKey(kmsClient, keyDesc));
        kmsClient.close();
    }

    public static String createKey(KmsClient kmsClient, String keyDesc) {
        try {
            CreateKeyRequest keyRequest = CreateKeyRequest.builder()
                .description(keyDesc)
                .customerMasterKeySpec(CustomerMasterKeySpec.SYMMETRIC_DEFAULT)
                .keyUsage("ENCRYPT_DECRYPT")
                .build();

            CreateKeyResponse result = kmsClient.createKey(keyRequest);
            System.out.printf("Created a customer key with id \"%s\"%n",
result.keyMetadata().arn());
            return result.keyMetadata().keyId();

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

```
}
```

- Weitere API-Informationen finden Sie unter [CreateKey](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen eines Alias für einen Schlüssel

Das folgende Codebeispiel zeigt, wie Sie einen Alias für einen KMS-Schlüssel erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.CreateAliasRequest;
import software.amazon.awssdk.services.kms.model.KmsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAlias {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <targetKeyId> <aliasName>\s

            Where:
                targetKeyId - The key ID or the Amazon Resource Name (ARN) of
                the customer master key (CMK).\s
```

```
        aliasName - An alias name (for example, alias/myAlias).\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String targetKeyId = args[0];
    String aliasName = args[1];
    Region region = Region.US_WEST_2;
    KmsClient kmsClient = KmsClient.builder()
        .region(region)
        .build();

    createCustomAlias(kmsClient, targetKeyId, aliasName);
    kmsClient.close();
}

public static void createCustomAlias(KmsClient kmsClient, String targetKeyId,
String aliasName) {
    try {
        CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
            .aliasName(aliasName)
            .targetKeyId(targetKeyId)
            .build();

        kmsClient.createAlias(aliasRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [CreateAlias](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Entschlüsseln von Geheimtext

Das folgende Codebeispiel zeigt, wie Geheimtext entschlüsselt wird, der mit einem KMS-Schlüssel verschlüsselt wurde.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void decryptData(KmsClient kmsClient, SdkBytes encryptedData,
String keyId) {
    try {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        DecryptResponse decryptResponse = kmsClient.decrypt(decryptRequest);
        decryptResponse.plaintext();

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [Entschlüsseln](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beschreiben eines Schlüssels

Das folgende Codebeispiel zeigt, wie Sie einen KMS-Schlüssel beschreiben.



## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.DescribeKeyRequest;
import software.amazon.awssdk.services.kms.model.DescribeKeyResponse;
import software.amazon.awssdk.services.kms.model.KmsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeKey {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <keyId>\s

                Where:
                keyId - A key id value to describe (for example,
                xxxxxbcd-12ab-34cd-56ef-1234567890ab).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String keyId = args[0];
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
```

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

    describeSpecifcKey(kmsClient, keyId);
    kmsClient.close();
}

public static void describeSpecifcKey(KmsClient kmsClient, String keyId) {
    try {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        DescribeKeyResponse response = kmsClient.describeKey(keyRequest);
        System.out.println("The key description is " +
response.keyMetadata().description());
        System.out.println("The key ARN is " + response.keyMetadata().arn());

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [DescribeKey](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Deaktivieren eines Schlüssels

Das folgende Codebeispiel zeigt, wie Sie einen KMS-Schlüssel deaktivieren.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.DisableKeyRequest;
import software.amazon.awssdk.services.kms.model.KmsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DisableCustomerKey {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <keyId>\s

            Where:
                keyId - A key id value to disable (for example,
                xxxxxbcd-12ab-34cd-56ef-1234567890ab).\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String keyId = args[0];
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        disableKey(kmsClient, keyId);
        kmsClient.close();
    }

    public static void disableKey(KmsClient kmsClient, String keyId) {
        try {
            DisableKeyRequest keyRequest = DisableKeyRequest.builder()
                .keyId(keyId)
                .build();
        }
    }
}
```

```
        kmsClient.disableKey(keyRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [DisableKey](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Aktivieren eines Schlüssels

Das folgende Codebeispiel zeigt, wie Sie einen KMS-Schlüssel aktivieren.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.EnableKeyRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EnableCustomerKey {
    public static void main(String[] args) {
```

```

    final String usage = ""

        Usage:
            <keyId>\s

        Where:
            keyId - A key id value to enable (for example,
xxxxxbcd-12ab-34cd-56ef-1234567890ab).\s
            """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String keyId = args[0];
    Region region = Region.US_WEST_2;
    KmsClient kmsClient = KmsClient.builder()
        .region(region)
        .build();

    enableKey(kmsClient, keyId);
    kmsClient.close();
}

public static void enableKey(KmsClient kmsClient, String keyId) {
    try {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.enableKey(enableKeyRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}

```

- Weitere API-Informationen finden Sie unter [EnableKey](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Verschlüsseln von Text mit einem Schlüssel

Das folgende Codebeispiel zeigt, wie Sie Text mit einem KMS-Schlüssel verschlüsseln.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.EncryptRequest;
import software.amazon.awssdk.services.kms.model.EncryptResponse;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.DecryptRequest;
import software.amazon.awssdk.services.kms.model.DecryptResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EncryptDataKey {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <keyId>\s

                Where:
                keyId - A key id value to use to encrypt/decrypt the data (for
                example, xxxxxbcd-12ab-34cd-56ef-1234567890ab).\s

                """;

        if (args.length != 1) {
```

```
        System.out.println(usage);
        System.exit(1);
    }

    String keyId = args[0];
    Region region = Region.US_WEST_2;
    KmsClient kmsClient = KmsClient.builder()
        .region(region)
        .build();

    SdkBytes encryData = encryptData(kmsClient, keyId);
    decryptData(kmsClient, encryData, keyId);
    System.out.println("Done");
    kmsClient.close();
}

public static SdkBytes encryptData(KmsClient kmsClient, String keyId) {
    try {
        SdkBytes myBytes = SdkBytes.fromByteArray(new byte[] { 1, 2, 3, 4, 5, 6,
7, 8, 9, 0 });
        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        EncryptResponse response = kmsClient.encrypt(encryptRequest);
        String algorithm = response.encryptionAlgorithm().toString();
        System.out.println("The encryption algorithm is " + algorithm);

        // Get the encrypted data.
        SdkBytes encryptedData = response.ciphertextBlob();
        return encryptedData;

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return null;
}

public static void decryptData(KmsClient kmsClient, SdkBytes encryptedData,
String keyId) {
    try {
        DecryptRequest decryptRequest = DecryptRequest.builder()
```

```

        .ciphertextBlob(encryptedData)
        .keyId(keyId)
        .build();

    DecryptResponse decryptResponse = kmsClient.decrypt(decryptRequest);
    decryptResponse.plaintext();

} catch (KmsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
}

```

- Weitere API-Informationen finden Sie unter [Verschlüsselung](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Aliassen für einen Schlüssel

Das folgende Codebeispiel zeigt, wie Sie Aliase für einen KMS-Schlüssel auflisten.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.AliasListEntry;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.ListAliasesRequest;
import software.amazon.awssdk.services.kms.model.ListAliasesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.

```



```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ListAliases {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        listAllAliases(kmsClient);
        kmsClient.close();
    }

    public static void listAllAliases(KmsClient kmsClient) {
        try {
            ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
                .limit(15)
                .build();

            ListAliasesResponse aliasesResponse =
kmsClient.listAliases(aliasesRequest);
            List<AliasListEntry> aliases = aliasesResponse.aliases();
            for (AliasListEntry alias : aliases) {
                System.out.println("The alias name is: " + alias.aliasName());
            }

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [ListAliases](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Erteilungen für einen Schlüssel

Das folgende Codebeispiel zeigt, wie Sie Erteilungen für einen KMS-Schlüssel auflisten.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.GrantListEntry;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.ListGrantsRequest;
import software.amazon.awssdk.services.kms.model.ListGrantsResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListGrants {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <keyId>\s

                Where:
                keyId - a key id value to use (for example,
                xxxxbcd-12ab-34cd-56ef-1234567890ab).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String keyId = args[0];
```

```
    Region region = Region.US_WEST_2;
    KmsClient kmsClient = KmsClient.builder()
        .region(region)
        .build();

    displayGrantIds(kmsClient, keyId);
    kmsClient.close();
}

public static void displayGrantIds(KmsClient kmsClient, String keyId) {
    try {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();

        ListGrantsResponse response = kmsClient.listGrants(grantsRequest);
        List<GrantListEntry> grants = response.grants();
        for (GrantListEntry grant : grants) {
            System.out.println("The grant Id is : " + grant.grantId());
        }

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [ListGrants](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Schlüsseln

Das folgende Codebeispiel zeigt, wie Sie KMS-Schlüssel auflisten.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.KeyListEntry;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.model.ListKeysResponse;
import software.amazon.awssdk.services.kms.model.KmsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListKeys {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        listAllKeys(kmsClient);
        kmsClient.close();
    }

    public static void listAllKeys(KmsClient kmsClient) {
        try {
            ListKeysRequest listKeysRequest = ListKeysRequest.builder()
                .limit(15)
                .build();

            ListKeysResponse keysResponse = kmsClient.listKeys(listKeysRequest);
        }
    }
}
```

```
        List<KeyListEntry> keyListEntries = keysResponse.keys();
        for (KeyListEntry key : keyListEntries) {
            System.out.println("The key ARN is: " + key.keyArn());
            System.out.println("The key Id is: " + key.keyId());
        }

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [ListKeys](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Lambda-Beispiele mit SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Lambda verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Erste Schritte

#### Hallo Lambda

Die folgenden Codebeispiele veranschaulichen, wie Sie mit der Verwendung von Lambda beginnen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
package com.example.lambda;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.ListFunctionsResponse;
import software.amazon.awssdk.services.lambda.model.FunctionConfiguration;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListLambdaFunctions {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        LambdaClient awsLambda = LambdaClient.builder()
            .region(region)
            .build();

        listFunctions(awsLambda);
        awsLambda.close();
    }

    public static void listFunctions(LambdaClient awsLambda) {
        try {
            ListFunctionsResponse functionResult = awsLambda.listFunctions();
            List<FunctionConfiguration> list = functionResult.functions();
            for (FunctionConfiguration config : list) {
                System.out.println("The function name is " + config.functionName());
            }
        }
    }
}
```

```
    }  
  
    } catch (LambdaException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}  
}
```

- Weitere API-Informationen finden Sie unter [ListFunctions](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Themen

- [Aktionen](#)
- [Szenarien](#)
- [Serverless-Beispiele](#)

## Aktionen

### Erstellen einer -Funktion

Das folgende Codebeispiel zeigt, wie Sie eine Lambda-Funktion erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;  
import software.amazon.awssdk.core.waiters.WaiterResponse;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.lambda.LambdaClient;  
import software.amazon.awssdk.services.lambda.model.CreateFunctionRequest;  
import software.amazon.awssdk.services.lambda.model.FunctionCode;  
import software.amazon.awssdk.services.lambda.model.CreateFunctionResponse;
```

```
import software.amazon.awssdk.services.lambda.model.GetFunctionRequest;
import software.amazon.awssdk.services.lambda.model.GetFunctionResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.Runtime;
import software.amazon.awssdk.services.lambda.waiters.LambdaWaiter;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;

/**
 * This code example requires a ZIP or JAR that represents the code of the
 * Lambda function.
 * If you do not have a ZIP or JAR, please refer to the following document:
 *
 * https://github.com/aws-doc-sdk-examples/tree/master/javav2/usecases/creating\_workflows\_stepfunctions
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CreateFunction {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <functionName> <filePath> <role> <handler>\s

            Where:
                functionName - The name of the Lambda function.\s
                filePath - The path to the ZIP or JAR where the code is located.
\s

                role - The role ARN that has Lambda permissions.\s
                handler - The fully qualified method name (for example,
example.Handler::handleRequest). \s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```



```
    }

    String functionName = args[0];
    String filePath = args[1];
    String role = args[2];
    String handler = args[3];
    Region region = Region.US_WEST_2;
    LambdaClient awsLambda = LambdaClient.builder()
        .region(region)
        .build();

    createLambdaFunction(awsLambda, functionName, filePath, role, handler);
    awsLambda.close();
}

public static void createLambdaFunction(LambdaClient awsLambda,
    String functionName,
    String filePath,
    String role,
    String handler) {

    try {
        LambdaWaiter waiter = awsLambda.waiter();
        InputStream is = new FileInputStream(filePath);
        SdkBytes fileToUpload = SdkBytes.fromInputStream(is);

        FunctionCode code = FunctionCode.builder()
            .zipFile(fileToUpload)
            .build();

        CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
            .functionName(functionName)
            .description("Created by the Lambda Java API")
            .code(code)
            .handler(handler)
            .runtime(Runtime.JAVA8)
            .role(role)
            .build();

        // Create a Lambda function using a waiter.
        CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
        GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
```

```

        .build();
        WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitForFunctionExists(getFunctionRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("The function ARN is " +
functionResponse.functionArn());

        } catch (LambdaException | FileNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}

```

- Weitere API-Informationen finden Sie unter [CreateFunction](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen einer -Funktion

Das folgende Codebeispiel zeigt, wie Sie eine Lambda-Funktion löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.DeleteFunctionRequest;
import software.amazon.awssdk.services.lambda.model.LambdaException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html

```

```
*/
public class DeleteFunction {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <functionName>\s

            Where:
                functionName - The name of the Lambda function.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        Region region = Region.US_EAST_1;
        LambdaClient awsLambda = LambdaClient.builder()
            .region(region)
            .build();

        deleteLambdaFunction(awsLambda, functionName);
        awsLambda.close();
    }

    public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                .functionName(functionName)
                .build();

            awsLambda.deleteFunction(request);
            System.out.println("The " + functionName + " function was deleted");

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteFunction](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Aufruf einer -Funktion

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion aufgerufen wird.

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import org.json.JSONObject;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.InvokeRequest;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.lambda.model.InvokeResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;

public class LambdaInvoke {

    /*
     * Function names appear as
     * arn:aws:lambda:us-west-2:335556666777:function:HelloFunction
     * you can retrieve the value by looking at the function in the AWS Console
     *
     * Also, set up your development environment, including your credentials.
     *
     * For information, see this documentation topic:
     *
     * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.
     * html
     */

    public static void main(String[] args) {
```

```
final String usage = ""

    Usage:
        <functionName>\s

    Where:
        functionName - The name of the Lambda function\s
    """;

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String functionName = args[0];
Region region = Region.US_WEST_2;
LambdaClient awsLambda = LambdaClient.builder()
    .region(region)
    .build();

invokeFunction(awsLambda, functionName);
awsLambda.close();
}

public static void invokeFunction(LambdaClient awsLambda, String functionName) {

    InvokeResponse res = null;
    try {
        // Need a SdkBytes instance for the payload.
        JSONObject jsonObj = new JSONObject();
        jsonObj.put("inputValue", "2000");
        String json = jsonObj.toString();
        SdkBytes payload = SdkBytes.fromUtf8String(json);

        // Setup an InvokeRequest.
        InvokeRequest request = InvokeRequest.builder()
            .functionName(functionName)
            .payload(payload)
            .build();

        res = awsLambda.invoke(request);
        String value = res.payload().asUtf8String();
        System.out.println(value);
    }
}
```

```
        } catch (LambdaException e) {  
            System.err.println(e.getMessage());  
            System.exit(1);  
        }  
    }  
}
```

- Weitere API-Informationen finden Sie unter [Invoke](#) in der AWS SDK for Java 2.x -API-Referenz.

## Szenarien

### Erste Schritte mit Funktionen

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 Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/*  
 * Lambda function names appear as:  
 *  
 * arn:aws:lambda:us-west-2:335556666777:function:HelloFunction  
 *
```

```
* To find this value, look at the function in the AWS Management Console.
*
* Before running this Java code example, set up your development environment,
including your credentials.
*
* For more information, see this documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* This example performs the following tasks:
*
* 1. Creates an AWS Lambda function.
* 2. Gets a specific AWS Lambda function.
* 3. Lists all Lambda functions.
* 4. Invokes a Lambda function.
* 5. Updates the Lambda function code and invokes it again.
* 6. Updates a Lambda function's configuration value.
* 7. Deletes a Lambda function.
*/
```

```
public class LambdaScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

            Usage:
                <functionName> <filePath> <role> <handler> <bucketName> <key>\s

            Where:
                functionName - The name of the Lambda function.\s
                filePath - The path to the .zip or .jar where the code is
located.\s
                role - The AWS Identity and Access Management (IAM) service role
that has Lambda permissions.\s
                handler - The fully qualified method name (for example,
example.Handler::handleRequest).\s
                bucketName - The Amazon Simple Storage Service (Amazon S3)
bucket name that contains the .zip or .jar used to update the Lambda function's
code.\s
                key - The Amazon S3 key name that represents the .zip or .jar
(for example, LambdaHello-1.0-SNAPSHOT.jar).
            """;
```

```
    if (args.length != 6) {
        System.out.println(usage);
        System.exit(1);
    }

    String functionName = args[0];
    String filePath = args[1];
    String role = args[2];
    String handler = args[3];
    String bucketName = args[4];
    String key = args[5];

    Region region = Region.US_WEST_2;
    LambdaClient awsLambda = LambdaClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the AWS Lambda example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Create an AWS Lambda function.");
    String funArn = createLambdaFunction(awsLambda, functionName, filePath,
role, handler);
    System.out.println("The AWS Lambda ARN is " + funArn);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Get the " + functionName + " AWS Lambda function.");
    getFunction(awsLambda, functionName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. List all AWS Lambda functions.");
    listFunctions(awsLambda);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. Invoke the Lambda function.");
    System.out.println("*** Sleep for 1 min to get Lambda function ready.");
    Thread.sleep(60000);
    invokeFunction(awsLambda, functionName);
    System.out.println(DASHES);
```



```
        System.out.println(DASHES);
        System.out.println("5. Update the Lambda function code and invoke it
again.");
        updateFunctionCode(awsLambda, functionName, bucketName, key);
        System.out.println("*** Sleep for 1 min to get Lambda function ready.");
        Thread.sleep(60000);
        invokeFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Update a Lambda function's configuration value.");
        updateFunctionConfiguration(awsLambda, functionName, handler);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the AWS Lambda function.");
        LambdaScenario.deleteLambdaFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS Lambda scenario completed successfully");
        System.out.println(DASHES);
        awsLambda.close();
    }

    public static String createLambdaFunction(LambdaClient awsLambda,
        String functionName,
        String filePath,
        String role,
        String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            InputStream is = new FileInputStream(filePath);
            SdkBytes fileToUpload = SdkBytes.fromInputStream(is);

            FunctionCode code = FunctionCode.builder()
                .zipFile(fileToUpload)
                .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                .functionName(functionName)
                .description("Created by the Lambda Java API")
```

```
        .code(code)
        .handler(handler)
        .runtime(Runtime.JAVA8)
        .role(role)
        .build();

    // Create a Lambda function using a waiter
    CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
    GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
        .functionName(functionName)
        .build();
    WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    return functionResponse.functionArn();

} catch (LambdaException | FileNotFoundException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}

public static void getFunction(LambdaClient awsLambda, String functionName) {
    try {
        GetFunctionRequest functionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();

        GetFunctionResponse response = awsLambda.getFunction(functionRequest);
        System.out.println("The runtime of this Lambda function is " +
response.configuration().runtime());

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listFunctions(LambdaClient awsLambda) {
    try {
        ListFunctionsResponse functionResult = awsLambda.listFunctions();
        List<FunctionConfiguration> list = functionResult.functions();
    }
}
```

```
        for (FunctionConfiguration config : list) {
            System.out.println("The function name is " + config.functionName());
        }

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void invokeFunction(LambdaClient awsLambda, String functionName) {

    InvokeResponse res;
    try {
        // Need a SdkBytes instance for the payload.
        JSONObject jsonObj = new JSONObject();
        jsonObj.put("inputValue", "2000");
        String json = jsonObj.toString();
        SdkBytes payload = SdkBytes.fromUtf8String(json);

        InvokeRequest request = InvokeRequest.builder()
            .functionName(functionName)
            .payload(payload)
            .build();

        res = awsLambda.invoke(request);
        String value = res.payload().asUtf8String();
        System.out.println(value);

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateFunctionCode(LambdaClient awsLambda, String
functionName, String bucketName, String key) {
    try {
        LambdaWaiter waiter = awsLambda.waiter();
        UpdateFunctionCodeRequest functionCodeRequest =
UpdateFunctionCodeRequest.builder()
            .functionName(functionName)
            .publish(true)
            .s3Bucket(bucketName)
```

```
        .s3Key(key)
        .build();

        UpdateFunctionCodeResponse response =
awsLambda.updateFunctionCode(functionCodeRequest);
        GetFunctionConfigurationRequest getFunctionConfigRequest =
GetFunctionConfigurationRequest.builder()
        .functionName(functionName)
        .build();

        WaiterResponse<GetFunctionConfigurationResponse> waiterResponse = waiter
        .waitUntilFunctionUpdated(getFunctionConfigRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("The last modified value is " +
response.lastModified());

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

    public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler) {
        try {
            UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
                .functionName(functionName)
                .handler(handler)
                .runtime(Runtime.JAVA11)
                .build();

            awsLambda.updateFunctionConfiguration(configurationRequest);

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
```

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

        awsLambda.deleteFunction(request);
        System.out.println("The " + functionName + " function was deleted");

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [CreateFunction](#)
  - [DeleteFunction](#)
  - [GetFunction](#)
  - [Aufrufen](#)
  - [ListFunctions](#)
  - [UpdateFunctionCode](#)
  - [UpdateFunctionConfiguration](#)

## Serverless-Beispiele

### Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöst wird. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

## Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Java.

```
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;

public class Handler implements RequestHandler<KinesisEvent, Void> {
    @Override
    public Void handleRequest(final KinesisEvent event, final Context context) {
        LambdaLogger logger = context.getLogger();
        if (event.getRecords().isEmpty()) {
            logger.log("Empty Kinesis Event received");
            return null;
        }
        for (KinesisEvent.KinesisEventRecord record : event.getRecords()) {
            try {
                logger.log("Processed Event with EventId: "+record.getEventID());
                String data = new String(record.getKinesis().getData().array());
                logger.log("Data:"+ data);
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex) {
                logger.log("An error occurred:"+ex.getMessage());
                throw ex;
            }
        }
        logger.log("Successfully processed:"+event.getRecords().size()+" records");
        return null;
    }
}
```

### Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

## Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Java

```
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
    com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotificat

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
            S3EventNotificationRecord record = s3event.getRecords().get(0);
            String srcBucket = record.getS3().getBucket().getName();
            String srcKey = record.getS3().getObject().getUrlDecodedKey();

            S3Client s3Client = S3Client.builder().build();
            HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
srcKey);

            logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
type " + headObject.contentType());

            return "Ok";
        } catch (Exception e) {
```

```
        throw new RuntimeException(e);
    }
}

private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
String key) {
    HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
        .bucket(bucket)
        .key(key)
        .build();
    return s3Client.headObject(headObjectRequest);
}
}
```

## Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten von einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

## Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Java.

```
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;

import java.util.Iterator;
```



```
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
        return Boolean.TRUE;
    }

    public void processRecord(SNSRecord record) {
        try {
            String message = record.getSNS().getMessage();
            logger.log("message: " + message);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}
```

## Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

## Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage;

public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
            context.getLogger().log("An error occurred");
            throw e;
        }
    }
}
```

## Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine partielle Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,
StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(KinesisEvent input, Context context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
ArrayList<>();
        String curRecordSequenceNumber = "";

        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :
input.getRecords()) {
            try {
                //Process your record
                KinesisEvent.Record kinesisRecord = kinesisEventRecord.getKinesis();
                curRecordSequenceNumber = kinesisRecord.getSequenceNumber();
            }
        }
    }
}
```

```
        } catch (Exception e) {
            /* Since we are working with streams, we can return the failed item
            immediately.
                Lambda will immediately begin to retry processing from this
            failed item onwards. */
            batchItemFailures.add(new
StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
            return new StreamsEventResponse(batchItemFailures);
        }
    }

    return new StreamsEventResponse(batchItemFailures);
}
}
```

## Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine partielle Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

## Melden von Fehlern bei SQS-Batchelementen mit Lambda unter Verwendung von Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;
```

```
public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {

        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
ArrayList<SQSBatchResponse.BatchItemFailure>();
        String messageId = "";
        for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
            try {
                //process your message
                messageId = message.getMessageId();
            } catch (Exception e) {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.add(new
SQSBatchResponse.BatchItemFailure(messageId));
            }
        }
        return new SQSBatchResponse(batchItemFailures);
    }
}
```

## MediaConvert -Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit verwenden MediaConvert.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Themen

- [Aktionen](#)

## Aktionen

### Erstellen eines Transcodierungsauftrags

Das folgende Codebeispiel zeigt, wie Sie einen AWS Elemental MediaConvert Transcodierungsauftrag erstellen.

SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
package com.example.mediaconvert;

import java.net.URI;
import java.util.HashMap;
import java.util.Map;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.Output;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroup;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.HlsGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupType;
import software.amazon.awssdk.services.mediaconvert.model.HlsDirectoryStructure;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestDurationFormat;
import software.amazon.awssdk.services.mediaconvert.model.HlsStreamInfResolution;
import software.amazon.awssdk.services.mediaconvert.model.HlsClientCache;
import software.amazon.awssdk.services.mediaconvert.model.HlsCaptionLanguageSetting;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestCompression;
import software.amazon.awssdk.services.mediaconvert.model.HlsCodecSpecification;
import software.amazon.awssdk.services.mediaconvert.model.HlsOutputSelection;
import software.amazon.awssdk.services.mediaconvert.model.HlsProgramDateTime;
import software.amazon.awssdk.services.mediaconvert.model.HlsTimedMetadataId3Frame;
import software.amazon.awssdk.services.mediaconvert.model.HlsSegmentControl;
import software.amazon.awssdk.services.mediaconvert.model.FileGroupSettings;
```

```
import software.amazon.awssdk.services.mediaconvert.model.ContainerSettings;
import software.amazon.awssdk.services.mediaconvert.model.VideoDescription;
import software.amazon.awssdk.services.mediaconvert.model.ContainerType;
import software.amazon.awssdk.services.mediaconvert.model.ScalingBehavior;
import software.amazon.awssdk.services.mediaconvert.model.VideoTimecodeInsertion;
import software.amazon.awssdk.services.mediaconvert.model.ColorMetadata;
import software.amazon.awssdk.services.mediaconvert.model.RespondToAfd;
import software.amazon.awssdk.services.mediaconvert.model.AfdSignaling;
import software.amazon.awssdk.services.mediaconvert.model.DropFrameTimecode;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264Settings;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodec;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.H264RateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.H264QualityTuningLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264SceneChangeDetect;
import
    software.amazon.awssdk.services.mediaconvert.model.AacAudioDescriptionBroadcasterMix;
import software.amazon.awssdk.services.mediaconvert.model.H264ParControl;
import software.amazon.awssdk.services.mediaconvert.model.AacRawFormat;
import software.amazon.awssdk.services.mediaconvert.model.H264QvbrSettings;
import
    software.amazon.awssdk.services.mediaconvert.model.H264FramerateConversionAlgorithm;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264FramerateControl;
import software.amazon.awssdk.services.mediaconvert.model.AacCodingMode;
import software.amazon.awssdk.services.mediaconvert.model.H264Telecine;
import
    software.amazon.awssdk.services.mediaconvert.model.H264FlickerAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264GopSizeUnits;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.H264GopBReference;
import software.amazon.awssdk.services.mediaconvert.model.AudioTypeControl;
import software.amazon.awssdk.services.mediaconvert.model.AntiAlias;
import software.amazon.awssdk.services.mediaconvert.model.H264SlowPal;
import
    software.amazon.awssdk.services.mediaconvert.model.H264SpatialAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264Syntax;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Settings;
import software.amazon.awssdk.services.mediaconvert.model.InputDenoiseFilter;
import
    software.amazon.awssdk.services.mediaconvert.model.H264TemporalAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobResponse;
```

```
import
software.amazon.awssdk.services.mediaconvert.model.H264UnregisteredSeiTimecode;
import software.amazon.awssdk.services.mediaconvert.model.H264EntropyEncoding;
import software.amazon.awssdk.services.mediaconvert.model.InputPsiControl;
import software.amazon.awssdk.services.mediaconvert.model.ColorSpace;
import software.amazon.awssdk.services.mediaconvert.model.H264RepeatPps;
import software.amazon.awssdk.services.mediaconvert.model.H264FieldEncoding;
import software.amazon.awssdk.services.mediaconvert.model.M3u8NielsenId3;
import software.amazon.awssdk.services.mediaconvert.model.InputDeblockFilter;
import software.amazon.awssdk.services.mediaconvert.model.InputRotate;
import software.amazon.awssdk.services.mediaconvert.model.H264DynamicSubGop;
import software.amazon.awssdk.services.mediaconvert.model.TimedMetadata;
import software.amazon.awssdk.services.mediaconvert.model.JobSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioDefaultSelection;
import software.amazon.awssdk.services.mediaconvert.model.VideoSelector;
import software.amazon.awssdk.services.mediaconvert.model.AacSpecification;
import software.amazon.awssdk.services.mediaconvert.model.Input;
import software.amazon.awssdk.services.mediaconvert.model.OutputSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264AdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.AudioLanguageCodeControl;
import software.amazon.awssdk.services.mediaconvert.model.InputFilterEnable;
import software.amazon.awssdk.services.mediaconvert.model.AudioDescription;
import software.amazon.awssdk.services.mediaconvert.model.H264InterlaceMode;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.AacSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodec;
import software.amazon.awssdk.services.mediaconvert.model.AacRateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.AacCodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.HlsIFrameOnlyManifest;
import software.amazon.awssdk.services.mediaconvert.model.FrameCaptureSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioSelector;
import software.amazon.awssdk.services.mediaconvert.model.M3u8PcrControl;
import software.amazon.awssdk.services.mediaconvert.model.InputTimecodeSource;
import software.amazon.awssdk.services.mediaconvert.model.HlsSettings;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Scte35Source;

/**
 * Create a MediaConvert job. Must supply MediaConvert access role Amazon
 * Resource Name (ARN), and a
 * valid video input file via Amazon S3 URL.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
```



```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
*/
public class CreateJob {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <mcRoleARN> <fileInput>\s

                Where:
                mcRoleARN - The MediaConvert Role ARN.\s
                fileInput - The URL of an Amazon S3 bucket
where the input file is located.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String mcRoleARN = args[0];
        String fileInput = args[1];
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
            .region(region)
            .build();

        String id = createMediaJob(mc, mcRoleARN, fileInput);
        System.out.println("MediaConvert job created. Job Id = " + id);
        mc.close();
    }

    public static String createMediaJob(MediaConvertClient mc, String mcRoleARN,
String fileInput) {

        String s3path = fileInput.substring(0, fileInput.lastIndexOf('/') +
1) + "javasdk/out/";
        String fileOutput = s3path + "index";
        String thumbsOutput = s3path + "thumbs/";
        String mp4Output = s3path + "mp4/";

        try {
```

```
DescribeEndpointsResponse res = mc

.describeEndpoints(DescribeEndpointsRequest.builder().maxResults(20).build());

    if (res.endpoints().size() <= 0) {
        System.out.println("Cannot find MediaConvert service
endpoint URL!");

        System.exit(1);
    }
String endpointURL = res.endpoints().get(0).url();
System.out.println("MediaConvert service URL: " +
endpointURL);

System.out.println("MediaConvert role arn: " + mcRoleARN);
System.out.println("MediaConvert input file: " + fileInput);
System.out.println("MediaConvert output path: " + s3path);

MediaConvertClient emc = MediaConvertClient.builder()
    .region(Region.US_WEST_2)
    .endpointOverride(URI.create(endpointURL))
    .build();

    // output group Preset HLS low profile
Output hlsLow = createOutput("hls_low", "_low", "_$dt$",
750000, 7, 1920, 1080, 640);
    // output group Preset HLS media profile
Output hlsMedium = createOutput("hls_medium", "_medium", "_
$dt$", 1200000, 7, 1920, 1080, 1280);
    // output group Preset HLS high profole
Output hlsHigh = createOutput("hls_high", "_high", "_$dt$",
3500000, 8, 1920, 1080, 1920);

    OutputGroup appleHLS = OutputGroup.builder().name("Apple
HLS").customName("Example")

.outputGroupSettings(OutputGroupSettings.builder()

.type(OutputGroupType.HLS_GROUP_SETTINGS)

.hlsGroupSettings(HlsGroupSettings.builder()

.directoryStructure(

    HlsDirectoryStructure.SINGLE_DIRECTORY)
```

```
.manifestDurationFormat(  
    HlsManifestDurationFormat.INTEGER)  
.streamInfResolution(  
    HlsStreamInfResolution.INCLUDE)  
.clientCache(HlsClientCache.ENABLED)  
.captionLanguageSetting(  
    HlsCaptionLanguageSetting.OMIT)  
.manifestCompression(  
    HlsManifestCompression.NONE)  
.codecSpecification(  
    HlsCodecSpecification.RFC_4281)  
.outputSelection(  
    HlsOutputSelection.MANIFESTS_AND_SEGMENTS)  
.programDateTime(HlsProgramDateTime.EXCLUDE)  
.programDateTimePeriod(600)  
.timedMetadataId3Frame(  
    HlsTimedMetadataId3Frame.PRIV)  
.timedMetadataId3Period(10)  
.destination(fileOutput)  
.segmentControl(HlsSegmentControl.SEGMENTED_FILES)  
.minFinalSegmentLength((double) 0)  
.segmentLength(4).minSegmentLength(0).build())
```

```
                .build())
                .outputs(hlsLow, hlsMedium,
hlsHigh).build());

        OutputGroup fileMp4 = OutputGroup.builder().name("File
Group").customName("mp4")

        .outputGroupSettings(OutputGroupSettings.builder()

        .type(OutputGroupType.FILE_GROUP_SETTINGS)

        .fileGroupSettings(FileGroupSettings.builder()

        .destination(mp4Output).build())

                .build())
                .outputs(Output.builder().extension("mp4")

        .containerSettings(ContainerSettings.builder()

        .container(ContainerType.MP4).build())

        .videoDescription(VideoDescription.builder().width(1280)

                .height(720)

        .scalingBehavior(ScalingBehavior.DEFAULT)

        .sharpness(50).antiAlias(AntiAlias.ENABLED)

        .timecodeInsertion(

                VideoTimecodeInsertion.DISABLED)

        .colorMetadata(ColorMetadata.INSERT)

        .respondToAfd(RespondToAfd.NONE)

        .afdSignaling(AfdSignaling.NONE)

        .dropFrameTimecode(DropFrameTimecode.ENABLED)

        .codecSettings(VideoCodecSettings.builder()

                .codec(VideoCodec.H_264)
```

```
.h264Settings(H264Settings

    .builder()

    .rateControlMode(

        H264RateControlMode.QVBR)

    .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)

    .qualityTuningLevel(

        H264QualityTuningLevel.SINGLE_PASS)

    .qvbrSettings(

        H264QvbrSettings.builder()

            .qvbrQualityLevel(

                8)

            .build())

    .codecLevel(H264CodecLevel.AUTO)

    .codecProfile(H264CodecProfile.MAIN)

    .maxBitrate(2400000)

    .framerateControl(

        H264FramerateControl.INITIALIZE_FROM_SOURCE)

    .gopSize(2.0)

    .gopSizeUnits(H264GopSizeUnits.SECONDS)

    .numberBFramesBetweenReferenceFrames(

        2)

    .gopClosedCadence(
```

1)

```
.gopBReference(H264GopBReference.DISABLED)
```

```
.slowPal(H264SlowPal.DISABLED)
```

```
.syntax(H264Syntax.DEFAULT)
```

```
.numberReferenceFrames(  

```

3)

```
.dynamicSubGop(H264DynamicSubGop.STATIC)
```

```
.fieldEncoding(H264FieldEncoding.PAFF)
```

```
.sceneChangeDetect(  

```

```
    H264SceneChangeDetect.ENABLED)
```

```
.minIInterval(0)
```

```
.telecine(H264Telecine.NONE)
```

```
.framerateConversionAlgorithm(  

```

```
    H264FramerateConversionAlgorithm.DUPLICATE_DROP)
```

```
.entropyEncoding(  

```

```
    H264EntropyEncoding.CABAC)
```

```
.slices(1)
```

```
.unregisteredSeiTimecode(  

```

```
    H264UnregisteredSeiTimecode.DISABLED)
```

```
.repeatPps(H264RepeatPps.DISABLED)
```

```
.adaptiveQuantization(  

```

```
    H264AdaptiveQuantization.HIGH)
```

```
        .spatialAdaptiveQuantization(
            H264SpatialAdaptiveQuantization.ENABLED)
        .temporalAdaptiveQuantization(
            H264TemporalAdaptiveQuantization.ENABLED)
        .flickerAdaptiveQuantization(
            H264FlickerAdaptiveQuantization.DISABLED)
        .softness(0)
        .interlaceMode(H264InterlaceMode.PROGRESSIVE)
        .build()

    .build()
                                                                    .build()

    .audioDescriptions(AudioDescription.builder()
        .audioTypeControl(AudioTypeControl.FOLLOW_INPUT)
        .languageCodeControl(
            AudioLanguageCodeControl.FOLLOW_INPUT)
        .codecSettings(AudioCodecSettings.builder()
            .codec(AudioCodec.AAC)
            .aacSettings(AacSettings
                .builder()
                    .codecProfile(AacCodecProfile.LC)
                    .rateControlMode(
                        AacRateControlMode.CBR)
```

```

        .codingMode(AacCodingMode.CODING_MODE_2_0)

        .sampleRate(44100)

        .bitrate(160000)

        .rawFormat(AacRawFormat.NONE)

        .specification(AacSpecification.MPEG4)

        .audioDescriptionBroadcasterMix(
            AacAudioDescriptionBroadcasterMix.NORMAL)

        .build()

    .build()

        .build()

        .build()

        .build();
        OutputGroup thumbs = OutputGroup.builder().name("File
Group").customName("thumbs")

        .outputGroupSettings(OutputGroupSettings.builder()

        .type(OutputGroupType.FILE_GROUP_SETTINGS)

        .fileGroupSettings(FileGroupSettings.builder()

        .destination(thumbsOutput).build()

        .build()

        .outputs(Output.builder().extension("jpg")

        .containerSettings(ContainerSettings.builder()

        .container(ContainerType.RAW).build()

        .videoDescription(VideoDescription.builder()

        .scalingBehavior(ScalingBehavior.DEFAULT)

        .sharpness(50).antiAlias(AntiAlias.ENABLED)

```



```

.timecodeInsertion(
    VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()

    .codec(VideoCodec.FRAME_CAPTURE)

    .frameCaptureSettings(
        FrameCaptureSettings
            .builder()

            .framerateNumerator(
                1)

            .framerateDenominator(
                1)

            .maxCaptures(10000000)

            .quality(80)

            .build())

    .build())

        .build())

            .build();

    Map<String, AudioSelector> audioSelectors = new HashMap<>();
    audioSelectors.put("Audio Selector 1",

AudioSelector.builder().defaultSelection(AudioDefaultSelection.DEFAULT)
    .offset(0).build());

```

```
        JobSettings jobSettings =
JobSettings.builder().inputs(Input.builder()
                                .audioSelectors(audioSelectors)
                                .videoSelector(

VideoSelector.builder().colorSpace(ColorSpace.FOLLOW)

.rotate(InputRotate.DEGREE_0).build())

.filterEnable(InputFilterEnable.AUTO).filterStrength(0)
                                .deblockFilter(InputDeblockFilter.DISABLED)

.denoiseFilter(InputDenoiseFilter.DISABLED).psiControl(InputPsiControl.USE_PSI)

.timecodeSource(InputTimecodeSource.EMBEDDED).fileInput(fileInput).build())
                                .outputGroups(appleHLS, thumbs,
fileMp4).build();

        CreateJobRequest createJobRequest =
CreateJobRequest.builder().role(mcRoleARN)
                                .settings(jobSettings)
                                .build();

        CreateJobResponse createJobResponse =
emc.createJob(createJobRequest);
        return createJobResponse.job().id();

    } catch (MediaConvertException e) {
        System.out.println(e.toString());
        System.exit(0);
    }
    return "";
}

private final static Output createOutput(String customName,
        String nameModifier,
        String segmentModifier,
        int qvbrMaxBitrate,
        int qvbrQualityLevel,
        int originWidth,
        int originHeight,
        int targetWidth) {
```

```

        int targetHeight = Math.round(originHeight * targetWidth /
originWidth)
                                - (Math.round(originHeight * targetWidth /
originWidth) % 4);
        Output output = null;
        try {
            output =
Output.builder().nameModifier(nameModifier).outputSettings(OutputSettings.builder()

.hlsSettings(HlsSettings.builder().segmentModifier(segmentModifier)

.audioGroupId("program_audio")

.iFrameOnlyManifest(HlsIFrameOnlyManifest.EXCLUDE).build())
                                .build())

.containerSettings(ContainerSettings.builder().container(ContainerType.M3_U8)

.m3u8Settings(M3u8Settings.builder().audioFramesPerPes(4)

.pcrControl(M3u8PcrControl.PCR_EVERY_PES_PACKET)

.pmtPid(480).privateMetadataPid(503)

.programNumber(1).patInterval(0).pmtInterval(0)

.scte35Source(M3u8Scte35Source.NONE)

.scte35Pid(500).nielsenId3(M3u8NielsenId3.NONE)

.timedMetadata(TimedMetadata.NONE)

.timedMetadataPid(502).videoPid(481)

.audioPids(482, 483, 484, 485, 486, 487, 488,

            489, 490, 491, 492)

                                .build())
                                .build()
                                .videoDescription(
VideoDescription.builder().width(targetWidth)

.height(targetHeight)

```

```
.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(
    VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

.respondToAfd(RespondToAfd.NONE)

.afdSignaling(AfdSignaling.NONE)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()
    .codec(VideoCodec.H_264)
    .h264Settings(H264Settings
        .builder()
        .rateControlMode(
            H264RateControlMode.QVBR)
        .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)
        .qualityTuningLevel(
            H264QualityTuningLevel.SINGLE_PASS)
        .qvbrSettings(H264QvbrSettings
            .builder()
            .qvbrQualityLevel(
                qvbrQualityLevel)
            .build())
        .build())
```

```
.codecLevel(H264CodecLevel.AUTO)

.codecProfile((targetHeight > 720
              && targetWidth > 1280)
              ? H264CodecProfile.HIGH
              : H264CodecProfile.MAIN)

.maxBitrate(qvbrMaxBitrate)

.framerateControl(
              H264FramerateControl.INITIALIZE_FROM_SOURCE)

.gopSize(2.0)

.gopSizeUnits(H264GopSizeUnits.SECONDS)

.numberBFramesBetweenReferenceFrames(
              2)

.gopClosedCadence(
              1)

.gopBReference(H264GopBReference.DISABLED)

.slowPal(H264SlowPal.DISABLED)

.syntax(H264Syntax.DEFAULT)

.numberReferenceFrames(
              3)

.dynamicSubGop(H264DynamicSubGop.STATIC)

.fieldEncoding(H264FieldEncoding.PAFF)

.sceneChangeDetect(
```

```
        H264SceneChangeDetect.ENABLED)

    .minIInterval(0)

    .telecine(H264Telecine.NONE)

    .framerateConversionAlgorithm(
        H264FramerateConversionAlgorithm.DUPLICATE_DROP)

    .entropyEncoding(
        H264EntropyEncoding.CABAC)

    .slices(1)

    .unregisteredSeiTimecode(
        H264UnregisteredSeiTimecode.DISABLED)

    .repeatPps(H264RepeatPps.DISABLED)

    .adaptiveQuantization(
        H264AdaptiveQuantization.HIGH)

    .spatialAdaptiveQuantization(
        H264SpatialAdaptiveQuantization.ENABLED)

    .temporalAdaptiveQuantization(
        H264TemporalAdaptiveQuantization.ENABLED)

    .flickerAdaptiveQuantization(
        H264FlickerAdaptiveQuantization.DISABLED)

    .softness(0)

    .interlaceMode(H264InterlaceMode.PROGRESSIVE)

    .build())
```

```

        .build()
                                                    .build()

        .audioDescriptions(AudioDescription.builder()

        .audioTypeControl(AudioTypeControl.FOLLOW_INPUT)

        .languageCodeControl(AudioLanguageCodeControl.FOLLOW_INPUT)

        .codecSettings(AudioCodecSettings.builder()

        .codec(AudioCodec.AAC).aacSettings(AacSettings

            .builder()

            .codecProfile(AacCodecProfile.LC)

            .rateControlMode(

                AacRateControlMode.CBR)

            .codingMode(AacCodingMode.CODING_MODE_2_0)

            .sampleRate(44100)

            .bitrate(96000)

            .rawFormat(AacRawFormat.NONE)

            .specification(AacSpecification.MPEG4)

            .audioDescriptionBroadcasterMix(

                AacAudioDescriptionBroadcasterMix.NORMAL)

            .build()

                                                    .build()

                                                    .build()

                .build();
        } catch (MediaConvertException e) {
            e.printStackTrace();
            System.exit(0);
        }
    }

```

```
        return output;
    }
}
```

- Weitere API-Informationen finden Sie unter [CreateJob](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Abrufen eines Transcodierungsauftrags

Das folgende Codebeispiel zeigt, wie Sie einen AWS Elemental MediaConvert Transcodierungsauftrag abrufen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.GetJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.GetJobResponse;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import java.net.URI;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetJob {

    public static void main(String[] args) {
```



```
final String usage = "\n" +
    " <jobId> \n\n" +
    "Where:\n" +
    " jobId - The job id value.\n\n";

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String jobId = args[0];
Region region = Region.US_WEST_2;
MediaConvertClient mc = MediaConvertClient.builder()
    .region(region)
    .build();

getSpecificJob(mc, jobId);
mc.close();
}

public static void getSpecificJob(MediaConvertClient mc, String jobId) {
    try {
        DescribeEndpointsResponse res =
mc.describeEndpoints(DescribeEndpointsRequest.builder()
            .maxResults(20)
            .build());

        if (res.endpoints().size() <= 0) {
            System.out.println("Cannot find MediaConvert service endpoint
URL!");
            System.exit(1);
        }
        String endpointURL = res.endpoints().get(0).url();
        MediaConvertClient emc = MediaConvertClient.builder()
            .region(Region.US_WEST_2)
            .endpointOverride(URI.create(endpointURL))
            .build();

        GetJobRequest jobRequest = GetJobRequest.builder()
            .id(jobId)
            .build();

        GetJobResponse response = emc.getJob(jobRequest);
```

```

        System.out.println("The ARN of the job is " + response.job().arn());
    } catch (MediaConvertException e) {
        System.out.println(e.toString());
        System.exit(0);
    }
}
}
}

```

- Weitere API-Informationen finden Sie unter [GetJob](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Auflisten von Transcodierungsaufträgen

Das folgende Codebeispiel zeigt, wie AWS Elemental MediaConvert Transcodierungsaufträge aufgelistet werden.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsRequest;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsResponse;
import software.amazon.awssdk.services.mediaconvert.model.Job;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import java.net.URI;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */

```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ListJobs {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
            .region(region)
            .build();

        listCompleteJobs(mc);
        mc.close();
    }

    public static void listCompleteJobs(MediaConvertClient mc) {
        try {
            DescribeEndpointsResponse res =
mc.describeEndpoints(DescribeEndpointsRequest.builder()
                .maxResults(20)
                .build());

            if (res.endpoints().size() <= 0) {
                System.out.println("Cannot find MediaConvert service endpoint
URL!");
                System.exit(1);
            }

            String endpointURL = res.endpoints().get(0).url();
            MediaConvertClient emc = MediaConvertClient.builder()
                .region(Region.US_WEST_2)
                .endpointOverride(URI.create(endpointURL))
                .build();

            ListJobsRequest jobsRequest = ListJobsRequest.builder()
                .maxResults(10)
                .status("COMPLETE")
                .build();

            ListJobsResponse jobsResponse = emc.listJobs(jobsRequest);
            List<Job> jobs = jobsResponse.jobs();
            for (Job job : jobs) {
                System.out.println("The JOB ARN is : " + job.arn());
            }
        }
    }
}
```

```
        } catch (MediaConvertException e) {  
            System.out.println(e.toString());  
            System.exit(0);  
        }  
    }  
}
```

- Weitere API-Informationen finden Sie unter [ListJobs](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Migration Hub-Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Migration Hub verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)

### Aktionen

#### Löschen eines Fortschrittsstreams

Das folgende Codebeispiel zeigt, wie Sie den Fortschrittsstream löschen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
    software.amazon.awssdk.services.migrationhub.model.DeleteProgressUpdateStreamRequest;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteProgressStream {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <progressStream>\s

                Where:
                progressStream - the name of a progress stream to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String progressStream = args[0];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
```

```
        .build();

        deleteStream(migrationClient, progressStream);
        migrationClient.close();
    }

    public static void deleteStream(MigrationHubClient migrationClient, String
streamName) {
        try {
            DeleteProgressUpdateStreamRequest deleteProgressUpdateStreamRequest =
DeleteProgressUpdateStreamRequest
                .builder()
                .progressUpdateStreamName(streamName)
                .build();

            migrationClient.deleteProgressUpdateStream(deleteProgressUpdateStreamRequest);
            System.out.println(streamName + " is deleted");

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteProgressUpdateStream](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Beschreiben des Migrationsstatus

Das folgende Codebeispiel zeigt, wie Sie den Migrationsstatus beschreiben.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
    software.amazon.awssdk.services.migrationhub.model.DescribeApplicationStateRequest;
import
    software.amazon.awssdk.services.migrationhub.model.DescribeApplicationStateResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAppState {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                DescribeAppState <appId>\s

                Where:
                appId - the application id value.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();

        describeApplicationState(migrationClient, appId);
        migrationClient.close();
    }
}
```

```
public static void describeApplicationState(MigrationHubClient migrationClient,
String appId) {
    try {
        DescribeApplicationStateRequest applicationStateRequest =
DescribeApplicationStateRequest.builder()
            .applicationId(appId)
            .build();

        DescribeApplicationStateResponse applicationStateResponse =
migrationClient
            .describeApplicationState(applicationStateRequest);
        System.out.println("The application status is " +
applicationStateResponse.applicationStatusAsString());

    } catch (MigrationHubException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [DescribeApplicationState](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Abrufen der Liste der mit einer Migration verknüpften Attribute

Das folgende Codebeispiel zeigt, wie Sie eine Liste der Attribute abrufen, die einer Migration zugeordnet sind.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
```



```
import
software.amazon.awssdk.services.migrationhub.model.DescribeMigrationTaskRequest;
import
software.amazon.awssdk.services.migrationhub.model.DescribeMigrationTaskResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeMigrationTask {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                DescribeMigrationTask <migrationTask> <progressStream>\s

            Where:
                migrationTask - the name of a migration task.\s
                progressStream - the name of a progress stream.\s
            """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String migrationTask = args[0];
        String progressStream = args[1];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();

        describeMigTask(migrationClient, migrationTask, progressStream);
        migrationClient.close();
    }
}
```

```
public static void describeMigTask(MigrationHubClient migrationClient, String
migrationTask,
    String progressStream) {
    try {
        DescribeMigrationTaskRequest migrationTaskRequestRequest =
DescribeMigrationTaskRequest.builder()
            .progressUpdateStream(progressStream)
            .migrationTaskName(migrationTask)
            .build();

        DescribeMigrationTaskResponse migrationTaskResponse = migrationClient
            .describeMigrationTask(migrationTaskRequestRequest);
        System.out.println("The name is " +
migrationTaskResponse.migrationTask().migrationTaskName());

    } catch (MigrationHubException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [DescribeMigrationTask](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Auflisten von Anwendungen

Das folgende Codebeispiel zeigt, wie Sie Anwendungen auflisten.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.ApplicationState;
```

```
import
    software.amazon.awssdk.services.migrationhub.model.ListApplicationStatesRequest;
import
    software.amazon.awssdk.services.migrationhub.model.ListApplicationStatesResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListApplications {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();

        listApps(migrationClient);
        migrationClient.close();
    }

    public static void listApps(MigrationHubClient migrationClient) {
        try {
            ListApplicationStatesRequest applicationStatesRequest =
                ListApplicationStatesRequest.builder()
                    .maxResults(10)
                    .build();

            ListApplicationStatesResponse response =
                migrationClient.listApplicationStates(applicationStatesRequest);
            List<ApplicationState> apps = response.applicationStateList();
            for (ApplicationState appState : apps) {
                System.out.println("App Id is " + appState.applicationId());
                System.out.println("The status is " +
                    appState.applicationStatus().toString());
            }
        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

```

        System.exit(1);
    }
}

```

- Weitere API-Informationen finden Sie unter [ListApplications](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von erstellten Artefakten

Das folgende Codebeispiel zeigt, wie erstellte Artefakte aufgelistet werden.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.CreatedArtifact;
import
    software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsRequest;
import
    software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * To run this Java V2 code example, ensure that you have setup your development
 * environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCreatedArtifacts {
    public static void main(String[] args) {

```

```
Region region = Region.US_WEST_2;
MigrationHubClient migrationClient = MigrationHubClient.builder()
    .region(region)
    .build();

listArtifacts(migrationClient);
migrationClient.close();
}

public static void listArtifacts(MigrationHubClient migrationClient) {
    try {
        ListCreatedArtifactsRequest listCreatedArtifactsRequest =
ListCreatedArtifactsRequest.builder()
        .maxResults(10)
        .migrationTaskName("SampleApp5")
        .progressUpdateStream("ProgressSteamB")
        .build();

        ListCreatedArtifactsResponse response =
migrationClient.listCreatedArtifacts(listCreatedArtifactsRequest);
        List<CreatedArtifact> apps = response.createdArtifactList();
        for (CreatedArtifact artifact : apps) {
            System.out.println("App Id is " + artifact.description());
            System.out.println("The name is " + artifact.name());
        }

    } catch (MigrationHubException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [ListCreatedArtifacts](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Auflisten von Migrationsaufgaben

Das folgende Codebeispiel zeigt, wie Migrationsaufgaben aufgelistet werden.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.ListMigrationTasksRequest;
import
    software.amazon.awssdk.services.migrationhub.model.ListMigrationTasksResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationTaskSummary;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListMigrationTasks {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();

        listMigrTasks(migrationClient);
        migrationClient.close();
    }

    public static void listMigrTasks(MigrationHubClient migrationClient) {
        try {
            ListMigrationTasksRequest listMigrationTasksRequest =
                ListMigrationTasksRequest.builder()
                    .maxResults(10)
                    .build();
```

```
        ListMigrationTasksResponse response =
migrationClient.listMigrationTasks(listMigrationTasksRequest);
        List<MigrationTaskSummary> migrationList =
response.migrationTaskSummaryList();
        for (MigrationTaskSummary migration : migrationList) {
            System.out.println("Migration task name is " +
migration.migrationTaskName());
            System.out.println("The Progress update stream is " +
migration.progressUpdateStream());
        }

    } catch (MigrationHubException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [ListMigrationTasks](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Registrieren einer Migrationsaufgabe

Das folgende Codebeispiel zeigt, wie Sie eine Migrationsaufgabe registrieren.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
software.amazon.awssdk.services.migrationhub.model.CreateProgressUpdateStreamRequest;
import
software.amazon.awssdk.services.migrationhub.model.ImportMigrationTaskRequest;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ImportMigrationTask {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <migrationTask> <progressStream>\s

            Where:
                migrationTask - the name of a migration task.\s
                progressStream - the name of a progress stream.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String migrationTask = args[0];
        String progressStream = args[1];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();

        importMigrTask(migrationClient, migrationTask, progressStream);
        migrationClient.close();
    }

    public static void importMigrTask(MigrationHubClient migrationClient, String
migrationTask, String progressStream) {
        try {
            CreateProgressUpdateStreamRequest progressUpdateStreamRequest =
CreateProgressUpdateStreamRequest.builder()
                .progressUpdateStreamName(progressStream)
                .dryRun(false)

```



```
        .build();

        migrationClient.createProgressUpdateStream(progressUpdateStreamRequest);
        ImportMigrationTaskRequest migrationTaskRequest =
ImportMigrationTaskRequest.builder()
        .migrationTaskName(migrationTask)
        .progressUpdateStream(progressStream)
        .dryRun(false)
        .build();

        migrationClient.importMigrationTask(migrationTaskRequest);

    } catch (MigrationHubException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [ImportMigrationTask](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Amazon-Personalize-Beispiele mit SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon Personalize verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Themen

- [Aktionen](#)

## Aktionen

### Erstellen eines Batch-Schnittstellenauftrags

Das folgende Codebeispiel zeigt, wie Sie einen Amazon-Personalize-Stapelschnittstellenauftrag erstellen.

SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createPersonalizeBatchInferenceJob(PersonalizeClient
personalizeClient,
                String solutionVersionArn,
                String jobName,
                String s3InputDataSourcePath,
                String s3DataDestinationPath,
                String roleArn,
                String explorationWeight,
                String explorationItemAgeCutOff) {

    long waitInMilliseconds = 60 * 1000;
    String status;
    String batchInferenceJobArn;

    try {

        // Set up data input and output parameters.
        S3DataConfig inputSource = S3DataConfig.builder()
            .path(s3InputDataSourcePath)
            .build();

        S3DataConfig outputDestination = S3DataConfig.builder()
            .path(s3DataDestinationPath)
            .build();

        BatchInferenceJobInput jobInput =
        BatchInferenceJobInput.builder()
```

```

        .s3DataSource(inputSource)
        .build();

        BatchInferenceJobOutput jobOutputLocation =
BatchInferenceJobOutput.builder()
        .s3DataDestination(outputDestination)
        .build();

        // Optional code to build the User-Personalization specific
item exploration
        // config.
        HashMap<String, String> explorationConfig = new HashMap<>();

        explorationConfig.put("explorationWeight",
explorationWeight);
        explorationConfig.put("explorationItemAgeCutOff",
explorationItemAgeCutOff);

        BatchInferenceJobConfig jobConfig =
BatchInferenceJobConfig.builder()
        .itemExplorationConfig(explorationConfig)
        .build();

        // End optional User-Personalization recipe specific code.

        CreateBatchInferenceJobRequest
createBatchInferenceJobRequest = CreateBatchInferenceJobRequest
        .builder()
        .solutionVersionArn(solutionVersionArn)
        .jobInput(jobInput)
        .jobOutput(jobOutputLocation)
        .jobName(jobName)
        .roleArn(roleArn)
        .batchInferenceJobConfig(jobConfig) //
Optional
        .build();

        batchInferenceJobArn =
personalizeClient.createBatchInferenceJob(createBatchInferenceJobRequest)
        .batchInferenceJobArn();

        DescribeBatchInferenceJobRequest
describeBatchInferenceJobRequest = DescribeBatchInferenceJobRequest
        .builder()

```

```

        .batchInferenceJobArn(batchInferenceJobArn)
        .build();

    long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;
    while (Instant.now().getEpochSecond() < maxTime) {

        BatchInferenceJob batchInferenceJob =
personalizeClient

        .describeBatchInferenceJob(describeBatchInferenceJobRequest)
            .batchInferenceJob();

        status = batchInferenceJob.status();
        System.out.println("Batch inference job status: " +
status);

        if (status.equals("ACTIVE") || status.equals("CREATE
FAILED")) {

            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
    return batchInferenceJobArn;

} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return "";
}

```

- Weitere API-Informationen finden Sie unter [CreateBatchInferenceJob](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erstellen einer Kampagne

Das folgende Codebeispiel zeigt, wie Sie eine Amazon-Personalize-Kampagne erstellen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void createPersonalCampaign(PersonalizeClient personalizeClient,
String solutionVersionArn,
    String name) {

    try {
        CreateCampaignRequest createCampaignRequest =
CreateCampaignRequest.builder()
            .minProvisionedTPS(1)
            .solutionVersionArn(solutionVersionArn)
            .name(name)
            .build();

        CreateCampaignResponse campaignResponse =
personalizeClient.createCampaign(createCampaignRequest);
        System.out.println("The campaign ARN is " +
campaignResponse.campaignArn());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [CreateCampaign](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen eines Datensatzes

Das folgende Codebeispiel zeigt, wie Sie einen Amazon-Personalize-Datensatz erstellen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createDataset(PersonalizeClient personalizeClient,
    String datasetName,
    String datasetGroupArn,
    String datasetType,
    String schemaArn) {
    try {
        CreateDatasetRequest request = CreateDatasetRequest.builder()
            .name(datasetName)
            .datasetGroupArn(datasetGroupArn)
            .datasetType(datasetType)
            .schemaArn(schemaArn)
            .build();

        String datasetArn = personalizeClient.createDataset(request)
            .datasetArn();
        System.out.println("Dataset " + datasetName + " created.");
        return datasetArn;

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Weitere API-Informationen finden Sie unter [CreateDataset](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen eines Datensatzexportauftrags

Das folgende Codebeispiel zeigt, wie Sie einen Amazon-Personalize-Datensatzexportauftrag erstellen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createDatasetExportJob(PersonalizeClient personalizeClient,
    String jobName,
    String datasetArn,
    IngestionMode ingestionMode,
    String roleArn,
    String s3BucketPath,
    String kmsKeyArn) {

    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String status = null;

    try {

        S3DataConfig exportS3DataConfig =
        S3DataConfig.builder().path(s3BucketPath).kmsKeyArn(kmsKeyArn).build();
        DatasetExportJobOutput jobOutput =
        DatasetExportJobOutput.builder().s3DataDestination(exportS3DataConfig)
            .build();

        CreateDatasetExportJobRequest createRequest =
        CreateDatasetExportJobRequest.builder()
            .jobName(jobName)
            .datasetArn(datasetArn)
            .ingestionMode(ingestionMode)
            .jobOutput(jobOutput)
            .roleArn(roleArn)
            .build();

        String datasetExportJobArn =
        personalizeClient.createDatasetExportJob(createRequest).datasetExportJobArn();

        DescribeDatasetExportJobRequest describeDatasetExportJobRequest =
        DescribeDatasetExportJobRequest.builder()
            .datasetExportJobArn(datasetExportJobArn)
```

```
        .build();

    long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

    while (Instant.now().getEpochSecond() < maxTime) {

        DatasetExportJob datasetExportJob = personalizeClient
            .describeDatasetExportJob(describeDatasetExportJobRequest)
            .datasetExportJob();

        status = datasetExportJob.status();
        System.out.println("Export job status: " + status);

        if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
            return status;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return "";
}
```

- Weitere API-Informationen finden Sie unter [CreateDatasetExportJob](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erstellen einer Datensatzgruppe

Das folgende Codebeispiel zeigt, wie Sie eine Amazon-Personalize-Datensatzgruppe erstellen.



## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createDatasetGroup(PersonalizeClient personalizeClient,
String datasetGroupName) {

    try {
        CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
                            .name(datasetGroupName)
                            .build();
        return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

Erstellen Sie eine Domain-Datensatzgruppe.

```
public static String createDomainDatasetGroup(PersonalizeClient
personalizeClient,
        String datasetGroupName,
        String domain) {

    try {
        CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
                            .name(datasetGroupName)
                            .domain(domain)
                            .build();
        return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
    }  
    return "";  
}
```

- Weitere API-Informationen finden Sie unter [CreateDatasetGroup](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erstellen eines Datensatz-Importauftrags

Das folgende Codebeispiel zeigt, wie Sie einen Amazon-Personalize-Datensatzimportauftrag erstellen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createPersonalizeDatasetImportJob(PersonalizeClient  
personalizeClient,  
    String jobName,  
    String datasetArn,  
    String s3BucketPath,  
    String roleArn) {  
  
    long waitInMilliseconds = 60 * 1000;  
    String status;  
    String datasetImportJobArn;  
  
    try {  
        DataSource importDataSource = DataSource.builder()  
            .dataLocation(s3BucketPath)  
            .build();  
  
        CreateDatasetImportJobRequest createDatasetImportJobRequest =  
        CreateDatasetImportJobRequest.builder()  
            .datasetArn(datasetArn)  
            .dataSource(importDataSource)
```

```
        .jobName(jobName)
        .roleArn(roleArn)
        .build();

    datasetImportJobArn =
personalizeClient.createDatasetImportJob(createDatasetImportJobRequest)
        .datasetImportJobArn();
    DescribeDatasetImportJobRequest describeDatasetImportJobRequest =
DescribeDatasetImportJobRequest.builder()
        .datasetImportJobArn(datasetImportJobArn)
        .build();

    long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

    while (Instant.now().getEpochSecond() < maxTime) {

        DatasetImportJob datasetImportJob = personalizeClient
            .describeDatasetImportJob(describeDatasetImportJobRequest)
            .datasetImportJob();

        status = datasetImportJob.status();
        System.out.println("Dataset import job status: " + status);

        if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
    return datasetImportJobArn;

} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return "";
}
```

- Weitere API-Informationen finden Sie unter [CreateDatasetImportJob](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erstellen eines Domänenschemas

Das folgende Codebeispiel zeigt, wie Sie ein Amazon-Personalize-Domainschema erstellen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createDomainSchema(PersonalizeClient personalizeClient,
String schemaName, String domain,
String filePath) {

String schema = null;
try {
    schema = new String(Files.readAllBytes(Paths.get(filePath)));
} catch (IOException e) {
    System.out.println(e.getMessage());
}

try {
    CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
        .name(schemaName)
        .domain(domain)
        .schema(schema)
        .build();

    String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

    System.out.println("Schema arn: " + schemaArn);

    return schemaArn;

} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
```

- Weitere API-Informationen finden Sie unter [CreateSchema](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen eines Filters

Das folgende Codebeispiel zeigt, wie Sie einen Amazon-Personalize-Filter erstellen.

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createFilter(PersonalizeClient personalizeClient,
    String filterName,
    String datasetGroupArn,
    String filterExpression) {
    try {
        CreateFilterRequest request = CreateFilterRequest.builder()
            .name(filterName)
            .datasetGroupArn(datasetGroupArn)
            .filterExpression(filterExpression)
            .build();

        return personalizeClient.createFilter(request).filterArn();
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Weitere API-Informationen finden Sie unter [CreateFilter](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen eines Empfehlens

Das folgende Codebeispiel zeigt, wie Sie einen Amazon-Personalize-Empfehlungsgeber erstellen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createRecommender(PersonalizeClient personalizeClient,
    String name,
    String datasetGroupArn,
    String recipeArn) {

    long maxTime = 0;
    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String recommenderStatus = "";

    try {
        CreateRecommenderRequest createRecommenderRequest =
CreateRecommenderRequest.builder()
            .datasetGroupArn(datasetGroupArn)
            .name(name)
            .recipeArn(recipeArn)
            .build();

        CreateRecommenderResponse recommenderResponse = personalizeClient
            .createRecommender(createRecommenderRequest);
        String recommenderArn = recommenderResponse.recommenderArn();
        System.out.println("The recommender ARN is " + recommenderArn);

        DescribeRecommenderRequest describeRecommenderRequest =
DescribeRecommenderRequest.builder()
            .recommenderArn(recommenderArn)
            .build();

        maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

        while (Instant.now().getEpochSecond() < maxTime) {
```

```

        recommenderStatus =
personalizeClient.describeRecommender(describeRecommenderRequest).recommender()
                .status();
        System.out.println("Recommender status: " + recommenderStatus);

        if (recommenderStatus.equals("ACTIVE") ||
recommenderStatus.equals("CREATE FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
    return recommenderArn;

} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

```

- Weitere API-Informationen finden Sie unter [CreateRecommender](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erstellen eines Schemas

Das folgende Codebeispiel zeigt, wie Sie ein Amazon-Personalize-Schema erstellen.

### SDK für Java 2.x

#### Note

Auf [GitHub](#) gibt es mehr Beispiele. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

public static String createSchema(PersonalizeClient personalizeClient, String
schemaName, String filePath) {

```

```
String schema = null;
try {
    schema = new String(Files.readAllBytes(Paths.get(filePath)));
} catch (IOException e) {
    System.out.println(e.getMessage());
}

try {
    CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
        .name(schemaName)
        .schema(schema)
        .build();

    String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

    System.out.println("Schema arn: " + schemaArn);

    return schemaArn;

} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
```

- Weitere API-Informationen finden Sie unter [CreateSchema](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen einer Lösung

Das folgende Codebeispiel zeigt, wie Sie eine Amazon-Personalize-Lösung erstellen.



## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createPersonalizeSolution(PersonalizeClient
personalizeClient,
        String datasetGroupArn,
        String solutionName,
        String recipeArn) {

    try {
        CreateSolutionRequest solutionRequest = CreateSolutionRequest.builder()
            .name(solutionName)
            .datasetGroupArn(datasetGroupArn)
            .recipeArn(recipeArn)
            .build();

        CreateSolutionResponse solutionResponse =
personalizeClient.createSolution(solutionRequest);
        return solutionResponse.solutionArn();

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Weitere API-Informationen finden Sie unter [CreateSolution](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen einer Lösungsversion

Das folgende Codebeispiel zeigt, wie Sie eine Amazon-Personalize-Lösung erstellen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createPersonalizeSolutionVersion(PersonalizeClient
personalizeClient, String solutionArn) {
    long maxTime = 0;
    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String solutionStatus = "";
    String solutionVersionStatus = "";
    String solutionVersionArn = "";

    try {
        DescribeSolutionRequest describeSolutionRequest =
DescribeSolutionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

        // Wait until solution is active.
        while (Instant.now().getEpochSecond() < maxTime) {

            solutionStatus =
personalizeClient.describeSolution(describeSolutionRequest).solution().status();
            System.out.println("Solution status: " + solutionStatus);

            if (solutionStatus.equals("ACTIVE") || solutionStatus.equals("CREATE
FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
    }
}
```

```
        if (solutionStatus.equals("ACTIVE")) {

            CreateSolutionVersionRequest createSolutionVersionRequest =
CreateSolutionVersionRequest.builder()
                .solutionArn(solutionArn)
                .build();

            CreateSolutionVersionResponse createSolutionVersionResponse =
personalizeClient
                .createSolutionVersion(createSolutionVersionRequest);
            solutionVersionArn =
createSolutionVersionResponse.solutionVersionArn();

            System.out.println("Solution version ARN: " + solutionVersionArn);

            DescribeSolutionVersionRequest describeSolutionVersionRequest =
DescribeSolutionVersionRequest.builder()
                .solutionVersionArn(solutionVersionArn)
                .build();

            while (Instant.now().getEpochSecond() < maxTime) {

                solutionVersionStatus =
personalizeClient.describeSolutionVersion(describeSolutionVersionRequest)
                    .solutionVersion().status();
                System.out.println("Solution version status: " +
solutionVersionStatus);

                if (solutionVersionStatus.equals("ACTIVE") ||
solutionVersionStatus.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return solutionVersionArn;
        }
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
        return "";  
    }
```

- Weitere API-Informationen finden Sie unter [CreateSolutionVersion](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erstellen eines Ereignis-Trackers

Das folgende Codebeispiel zeigt, wie Sie einen Amazon-Personalize-Ereignis-Tracker erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createEventTracker(PersonalizeClient personalizeClient,  
String eventTrackerName,  
    String datasetGroupArn) {  
  
    String eventTrackerId = "";  
    String eventTrackerArn;  
    long maxTime = 3 * 60 * 60; // 3 hours  
    long waitInMilliseconds = 20 * 1000; // 20 seconds  
    String status;  
  
    try {  
  
        CreateEventTrackerRequest createEventTrackerRequest =  
CreateEventTrackerRequest.builder()  
            .name(eventTrackerName)  
            .datasetGroupArn(datasetGroupArn)  
            .build();  
  
        CreateEventTrackerResponse createEventTrackerResponse =  
personalizeClient  
            .createEventTracker(createEventTrackerRequest);
```

```

    eventTrackerArn = createEventTrackerResponse.eventTrackerArn();
    eventTrackerId = createEventTrackerResponse.trackingId();
    System.out.println("Event tracker ARN: " + eventTrackerArn);
    System.out.println("Event tracker ID: " + eventTrackerId);

    maxTime = Instant.now().getEpochSecond() + maxTime;

    DescribeEventTrackerRequest describeRequest =
DescribeEventTrackerRequest.builder()
        .eventTrackerArn(eventTrackerArn)
        .build();

    while (Instant.now().getEpochSecond() < maxTime) {

        status =
personalizeClient.describeEventTracker(describeRequest).eventTracker().status();
        System.out.println("EventTracker status: " + status);

        if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
    return eventTrackerId;
} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return eventTrackerId;
}

```

- Weitere API-Informationen finden Sie unter [CreateEventTracker](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Löschen einer Kampagne

Das folgende Codebeispiel zeigt, wie Sie eine Kampagne in Amazon Personalize löschen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void deleteSpecificCampaign(PersonalizeClient personalizeClient,
String campaignArn) {

    try {
        DeleteCampaignRequest campaignRequest = DeleteCampaignRequest.builder()
            .campaignArn(campaignArn)
            .build();

        personalizeClient.deleteCampaign(campaignRequest);


    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteCampaign](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen einer Lösung

Das folgende Codebeispiel zeigt, wie Sie eine Lösung in Amazon Personalize löschen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void deleteGivenSolution(PersonalizeClient personalizeClient,
String solutionArn) {

    try {
        DeleteSolutionRequest solutionRequest = DeleteSolutionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        personalizeClient.deleteSolution(solutionRequest);
        System.out.println("Done");

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteSolution](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen eines Ereignis-Trackers

Das folgende Codebeispiel zeigt, wie Sie einen Ereignis-Tracker in Amazon Personalize löschen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void deleteEventTracker(PersonalizeClient personalizeClient,
String eventTrackerArn) {
    try {
        DeleteEventTrackerRequest deleteEventTrackerRequest =
DeleteEventTrackerRequest.builder()
            .eventTrackerArn(eventTrackerArn)
            .build();
```

```
        int status =
personalizeClient.deleteEventTracker(deleteEventTrackerRequest).sdkHttpResponse().statusCode();

        System.out.println("Status code:" + status);

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteEventTracker](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Beschreiben einer Kampagne

Das folgende Codebeispiel zeigt, wie Sie eine Kampagne in Amazon Personalize beschreiben.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void describeSpecificCampaign(PersonalizeClient personalizeClient,
String campaignArn) {

    try {
        DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
            .campaignArn(campaignArn)
            .build();

        DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
        Campaign myCampaign = campaignResponse.campaign();
        System.out.println("The Campaign name is " + myCampaign.name());
        System.out.println("The Campaign status is " + myCampaign.status());
    }
}
```



```
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Weitere API-Informationen finden Sie unter [DescribeCampaign](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Beschreiben eines Rezepts

Das folgende Codebeispiel zeigt, wie Sie ein Rezept in Amazon Personalize beschreiben.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void describeSpecificRecipe(PersonalizeClient personalizeClient,  
String recipeArn) {  
  
    try {  
        DescribeRecipeRequest recipeRequest = DescribeRecipeRequest.builder()  
            .recipeArn(recipeArn)  
            .build();  
  
        DescribeRecipeResponse recipeResponse =  
personalizeClient.describeRecipe(recipeRequest);  
        System.out.println("The recipe name is " +  
recipeResponse.recipe().name());  
  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Weitere API-Informationen finden Sie unter [DescribeRecipe](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beschreiben einer Lösung

Das folgende Codebeispiel zeigt, wie Sie eine Lösung in Amazon Personalize beschreiben.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void describeSpecificSolution(PersonalizeClient personalizeClient,
String solutionArn) {

    try {
        DescribeSolutionRequest solutionRequest =
DescribeSolutionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        DescribeSolutionResponse response =
personalizeClient.describeSolution(solutionRequest);
        System.out.println("The Solution name is " +
response.solution().name());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeSolution](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Kampagnen

Das folgende Codebeispiel zeigt, wie Kampagnen in Amazon Personalize aufgelistet werden.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void listAllCampaigns(PersonalizeClient personalizeClient, String
solutionArn) {

    try {
        ListCampaignsRequest campaignsRequest = ListCampaignsRequest.builder()
            .maxResults(10)
            .solutionArn(solutionArn)
            .build();

        ListCampaignsResponse response =
personalizeClient.listCampaigns(campaignsRequest);
        List<CampaignSummary> campaigns = response.campaigns();
        for (CampaignSummary campaign : campaigns) {
            System.out.println("Campaign name is : " + campaign.name());
            System.out.println("Campaign ARN is : " + campaign.campaignArn());
        }

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [ListCampaigns](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Datensatzgruppen

Das folgende Codebeispiel zeigt, wie Sie Datensatzgruppen in Amazon Personalize auflisten.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void listDSGroups(PersonalizeClient personalizeClient) {  
  
    try {  
        ListDatasetGroupsRequest groupsRequest =  
ListDatasetGroupsRequest.builder()  
            .maxResults(15)  
            .build();  
  
        ListDatasetGroupsResponse groupsResponse =  
personalizeClient.listDatasetGroups(groupsRequest);  
        List<DatasetGroupSummary> groups = groupsResponse.datasetGroups();  
        for (DatasetGroupSummary group : groups) {  
            System.out.println("The DataSet name is : " + group.name());  
            System.out.println("The DataSet ARN is : " +  
group.datasetGroupArn());  
        }  
  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Weitere API-Informationen finden Sie unter [ListDatasetGroups](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Auflisten von Rezepten

Das folgende Codebeispiel zeigt, wie Rezepte in Amazon Personalize aufgelistet werden.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void listAllRecipes(PersonalizeClient personalizeClient) {  
  
    try {  
        ListRecipesRequest recipesRequest = ListRecipesRequest.builder()  
            .maxResults(15)  
            .build();  
  
        ListRecipesResponse response =  
personalizeClient.listRecipes(recipesRequest);  
        List<RecipeSummary> recipes = response.recipes();  
        for (RecipeSummary recipe : recipes) {  
            System.out.println("The recipe ARN is: " + recipe.recipeArn());  
            System.out.println("The recipe name is: " + recipe.name());  
        }  
  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Weitere API-Informationen finden Sie unter [ListRecipes](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Lösungen auflisten

Das folgende Codebeispiel zeigt, wie Sie Lösungen in Amazon Personalize auflisten.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void listAllSolutions(PersonalizeClient personalizeClient, String
datasetGroupArn) {

    try {
        ListSolutionsRequest solutionsRequest = ListSolutionsRequest.builder()
            .maxResults(10)
            .datasetGroupArn(datasetGroupArn)
            .build();

        ListSolutionsResponse response =
personalizeClient.listSolutions(solutionsRequest);
        List<SolutionSummary> solutions = response.solutions();
        for (SolutionSummary solution : solutions) {
            System.out.println("The solution ARN is: " +
solution.solutionArn());
            System.out.println("The solution name is: " + solution.name());
        }

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [ListSolutions](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Aktualisieren einer Kampagne

Das folgende Codebeispiel zeigt, wie Sie eine Kampagne in Amazon Personalize aktualisieren.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String updateCampaign(PersonalizeClient personalizeClient,
    String campaignArn,
    String solutionVersionArn,
    Integer minProvisionedTPS) {

    try {
        // build the updateCampaignRequest
        UpdateCampaignRequest updateCampaignRequest =
UpdateCampaignRequest.builder()
            .campaignArn(campaignArn)
            .solutionVersionArn(solutionVersionArn)
            .minProvisionedTPS(minProvisionedTPS)
            .build();

        // update the campaign
        personalizeClient.updateCampaign(updateCampaignRequest);

        DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
            .campaignArn(campaignArn)
            .build();

        DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
        Campaign updatedCampaign = campaignResponse.campaign();

        System.out.println("The Campaign status is " +
updatedCampaign.status());
        return updatedCampaign.status();

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
        return "";  
    }
```

- Weitere API-Informationen finden Sie unter [UpdateCampaign](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beispiele für Amazon Personalize Events mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon Personalize Events verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)

## Aktionen

### Importieren von Echtzeit-Interaktionsereignisdaten

Das folgende Codebeispiel zeigt, wie Echtzeit-Interaktionsereignisdaten in Amazon Personalize Events importiert werden.

SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.



```
public static int putItems(PersonalizeEventsClient personalizeEventsClient,
    String datasetArn,
    String item1Id,
    String item1PropertyName,
    String item1PropertyValue,
    String item2Id,
    String item2PropertyName,
    String item2PropertyValue) {

    int responseCode = 0;
    ArrayList<Item> items = new ArrayList<>();

    try {
        Item item1 = Item.builder()
            .itemId(item1Id)
            .properties(String.format("{\ \"%1$s\": \"%2$s
\}"),
                item1PropertyName,
                item1PropertyValue))
            .build();

        items.add(item1);

        Item item2 = Item.builder()
            .itemId(item2Id)
            .properties(String.format("{\ \"%1$s\": \"%2$s
\}"),
                item2PropertyName,
                item2PropertyValue))
            .build();

        items.add(item2);

        PutItemsRequest putItemsRequest = PutItemsRequest.builder()
            .datasetArn(datasetArn)
            .items(items)
            .build();

        responseCode =
personalizeEventsClient.putItems(putItemsRequest).sdkHttpResponse().statusCode();
        System.out.println("Response code: " + responseCode);
        return responseCode;
    }
```

```

        } catch (PersonalizeEventsException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return responseCode;
    }
}

```

- Weitere API-Informationen finden Sie unter [PutEvents](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Inkrementeller Import eines Benutzers

Das folgende Codebeispiel zeigt, wie Sie einen Benutzer inkrementell in Amazon Personalize Events importieren.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

public static int putUsers(PersonalizeEventsClient personalizeEventsClient,
    String datasetArn,
    String user1Id,
    String user1PropertyName,
    String user1PropertyValue,
    String user2Id,
    String user2PropertyName,
    String user2PropertyValue) {

    int responseCode = 0;
    ArrayList<User> users = new ArrayList<>();

    try {
        User user1 = User.builder()
            .userId(user1Id)
            .properties(String.format("{\ \"%1$s\": \"%2$s
\ }"),

```

```

        user1PropertyName,
user1PropertyValue))
        .build();

        users.add(user1);

        User user2 = User.builder()
            .userId(user2Id)
            .properties(String.format("{\\"%1$s\\": \\"%2$s
\\"}"),
                user2PropertyName,
user2PropertyValue))
            .build();

        users.add(user2);

        PutUsersRequest putUsersRequest = PutUsersRequest.builder()
            .datasetArn(datasetArn)
            .users(users)
            .build();

        responseCode =
personalizeEventsClient.putUsers(putUsersRequest).sdkHttpResponse().statusCode();
        System.out.println("Response code: " + responseCode);
        return responseCode;

    } catch (PersonalizeEventsException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return responseCode;
}

```

- Weitere API-Informationen finden Sie unter [PutUsers](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beispiele für Amazon Personalize Runtime mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon Personalize Runtime verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)

## Aktionen

Abrufen von Empfehlungen (benutzerdefinierte Datensatzgruppe)

Das folgende Codebeispiel zeigt, wie Sie Empfehlungen mit Rang für Amazon Personalize Runtime abrufen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static List<PredictedItem> getRankedRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
        String campaignArn,
        String userId,
        ArrayList<String> items) {

    try {
        GetPersonalizedRankingRequest rankingRecommendationsRequest =
        GetPersonalizedRankingRequest.builder()
            .campaignArn(campaignArn)
            .userId(userId)
            .inputList(items)
```

```
        .build();


        GetPersonalizedRankingResponse recommendationsResponse =
personalizeRuntimeClient
            .getPersonalizedRanking(rankingRecommendationsRequest);
        List<PredictedItem> rankedItems =
recommendationsResponse.personalizedRanking();
        int rank = 1;
        for (PredictedItem item : rankedItems) {
            System.out.println("Item ranked at position " + rank + " details");
            System.out.println("Item Id is : " + item.itemId());
            System.out.println("Item score is : " + item.score());
            System.out.println("-----");
            rank++;
        }
        return rankedItems;
    } catch (PersonalizeRuntimeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
```

- Weitere API-Informationen finden Sie unter [GetPersonalizedRanking](#) in der APIAWS SDK for Java 2.x -Referenz für .

Holen Sie sich Empfehlungen von einem Empfehler (Domain-Datensatzgruppe)

Das folgende Codebeispiel zeigt, wie Sie Empfehlungen zur Laufzeit von Amazon Personalize erhalten.

SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Rufen Sie eine Liste der empfohlenen Elemente ab.

```
public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String campaignArn, String userId) {

    try {
        GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
            .campaignArn(campaignArn)
            .numResults(20)
            .userId(userId)
            .build();

        GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
            .getRecommendations(recommendationsRequest);
        List<PredictedItem> items = recommendationsResponse.itemList();
        for (PredictedItem item : items) {
            System.out.println("Item Id is : " + item.itemId());
            System.out.println("Item score is : " + item.score());
        }

    } catch (AwsServiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Rufen Sie eine Liste der empfohlenen Elemente von einem Empfehler ab, der in einer Domain-Datensatzgruppe erstellt wurde.

```
public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String recommenderArn,
    String userId) {

    try {
        GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
            .recommenderArn(recommenderArn)
            .numResults(20)
            .userId(userId)
            .build();
```

```

        GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
            .getRecommendations(recommendationsRequest);
        List<PredictedItem> items = recommendationsResponse.itemList();

        for (PredictedItem item : items) {
            System.out.println("Item Id is : " + item.itemId());
            System.out.println("Item score is : " + item.score());
        }
    } catch (AwsServiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

Verwenden Sie einen Filter, wenn Sie Empfehlungen anfordern.

```

public static void getFilteredRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
    String campaignArn,
    String userId,
    String filterArn,
    String parameter1Name,
    String parameter1Value1,
    String parameter1Value2,
    String parameter2Name,
    String parameter2Value) {

    try {

        Map<String, String> filterValues = new HashMap<>();

        filterValues.put(parameter1Name, String.format("%1$s\", \"%2$s\"",
            parameter1Value1, parameter1Value2));
        filterValues.put(parameter2Name, String.format("%1$s\"",
            parameter2Value));

        GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
            .campaignArn(campaignArn)
            .numResults(20)
            .userId(userId)

```

```
        .filterArn(filterArn)
        .filterValues(filterValues)
        .build();

    GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
        .getRecommendations(recommendationsRequest);
    List<PredictedItem> items = recommendationsResponse.itemList();

    for (PredictedItem item : items) {
        System.out.println("Item Id is : " + item.itemId());
        System.out.println("Item score is : " + item.score());
    }
} catch (PersonalizeRuntimeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [GetRecommendations](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Amazon Pinpoint-Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon Pinpoint verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Themen

- [Aktionen](#)



## Aktionen

### Erstellen einer Kampagne

Das folgende Codebeispiel zeigt, wie Sie eine Kampagne erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Erstellen Sie eine Kampagne.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CampaignResponse;
import software.amazon.awssdk.services.pinpoint.model.Message;
import software.amazon.awssdk.services.pinpoint.model.Schedule;
import software.amazon.awssdk.services.pinpoint.model.Action;
import software.amazon.awssdk.services.pinpoint.model.MessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.WriteCampaignRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignResponse;
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCampaign {
    public static void main(String[] args) {

        final String usage = ""

            Usage:  <appId> <segmentId>
```

```
        Where:
            appId - The ID of the application to create the campaign in.
            segmentId - The ID of the segment to create the campaign from.
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String appId = args[0];
    String segmentId = args[1];
    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

    createPinCampaign(pinpoint, appId, segmentId);
    pinpoint.close();
}

public static void createPinCampaign(PinpointClient pinpoint, String appId,
String segmentId) {
    CampaignResponse result = createCampaign(pinpoint, appId, segmentId);
    System.out.println("Campaign " + result.name() + " created.");
    System.out.println(result.description());
}

public static CampaignResponse createCampaign(PinpointClient client, String
appID, String segmentID) {

    try {
        Schedule schedule = Schedule.builder()
            .startTime("IMMEDIATE")
            .build();

        Message defaultMessage = Message.builder()
            .action(Action.OPEN_APP)
            .body("My message body.")
            .title("My message title.")
            .build();

        MessageConfiguration messageConfiguration =
MessageConfiguration.builder()
            .defaultMessage(defaultMessage)
```

```
        .build();

        WriteCampaignRequest request = WriteCampaignRequest.builder()
            .description("My description")
            .schedule(schedule)
            .name("MyCampaign")
            .segmentId(segmentID)
            .messageConfiguration(messageConfiguration)
            .build();

        CreateCampaignResponse result =
client.createCampaign(CreateCampaignRequest.builder()
    .applicationId(appID)
    .writeCampaignRequest(request).build());

        System.out.println("Campaign ID: " + result.campaignResponse().id());
        return result.campaignResponse();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
}
```

- Weitere API-Informationen finden Sie unter [CreateCampaign](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen eines Segments

Das folgende Codebeispiel zeigt, wie Sie ein Segment erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.AttributeDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.AttributeType;
import software.amazon.awssdk.services.pinpoint.model.RecencyDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentBehaviors;
import software.amazon.awssdk.services.pinpoint.model.SegmentDemographics;
import software.amazon.awssdk.services.pinpoint.model.SegmentLocation;
import software.amazon.awssdk.services.pinpoint.model.SegmentDimensions;
import software.amazon.awssdk.services.pinpoint.model.WriteSegmentRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateSegment {
    public static void main(String[] args) {
        final String usage = ""

                Usage:  <appId>

                Where:
                    appId - The application ID to create a segment
for.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
```

```
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        SegmentResponse result = createSegment(pinpoint, appId);
        System.out.println("Segment " + result.name() + " created.");
        System.out.println(result.segmentType());
        pinpoint.close();
    }

    public static SegmentResponse createSegment(PinpointClient client, String
appId) {
        try {
            Map<String, AttributeDimension> segmentAttributes = new
HashMap<>();

            segmentAttributes.put("Team", AttributeDimension.builder()
                .attributeType(AttributeType.INCLUSIVE)
                .values("Lakers")
                .build());

            RecencyDimension recencyDimension =
RecencyDimension.builder()
                .duration("DAY_30")
                .recencyType("ACTIVE")
                .build();

            SegmentBehaviors segmentBehaviors =
SegmentBehaviors.builder()
                .recency(recencyDimension)
                .build();

            SegmentDemographics segmentDemographics =
SegmentDemographics
                .builder()
                .build();

            SegmentLocation segmentLocation = SegmentLocation
                .builder()
                .build();

            SegmentDimensions dimensions = SegmentDimensions
                .builder()
                .attributes(segmentAttributes)
                .behavior(segmentBehaviors)
```

```

        .demographic(segmentDemographics)
        .location(segmentLocation)
        .build();

        WriteSegmentRequest writeSegmentRequest =
WriteSegmentRequest.builder()
                    .name("MySegment")
                    .dimensions(dimensions)
                    .build();

        CreateSegmentRequest createSegmentRequest =
CreateSegmentRequest.builder()
                    .applicationId(appId)
                    .writeSegmentRequest(writeSegmentRequest)
                    .build();

        CreateSegmentResponse createSegmentResult =
client.createSegment(createSegmentRequest);
        System.out.println("Segment ID: " +
createSegmentResult.segmentResponse().id());
        System.out.println("Done");
        return createSegmentResult.segmentResponse();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
}

```

- Weitere API-Informationen finden Sie unter [CreateSegment](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen einer Anwendung

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateAppRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateAppResponse;
import software.amazon.awssdk.services.pinpoint.model.CreateApplicationRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateApp {
    public static void main(String[] args) {
        final String usage = ""

                Usage: <appName>

                Where:
                appName - The name of the application to create.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        String appName = args[0];
        System.out.println("Creating an application with name: " + appName);

        PinpointClient pinpoint = PinpointClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    String appID = createApplication(pinpoint, appName);
    System.out.println("App ID is: " + appID);
    pinpoint.close();
}

public static String createApplication(PinpointClient pinpoint, String appName)
{
    try {
        CreateApplicationRequest appRequest = CreateApplicationRequest.builder()
            .name(appName)
            .build();

        CreateAppRequest request = CreateAppRequest.builder()
            .createApplicationRequest(appRequest)
            .build();

        CreateAppResponse result = pinpoint.createApp(request);
        return result.applicationResponse().id();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Weitere API-Informationen finden Sie unter [CreateApp](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen einer Anwendung

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung löschen.



## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Löschen Sie eine Anwendung.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DeleteAppRequest;
import software.amazon.awssdk.services.pinpoint.model.DeleteAppResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteApp {
    public static void main(String[] args) {
        final String usage = ""

            Usage: <appId>

            Where:
            appId - The ID of the application to delete.

            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        System.out.println("Deleting an application with ID: " + appId);
        PinpointClient pinpoint = PinpointClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    deletePinApp(pinpoint, appId);
    System.out.println("Done");
    pinpoint.close();
}

public static void deletePinApp(PinpointClient pinpoint, String appId) {
    try {
        DeleteAppRequest appRequest = DeleteAppRequest.builder()
            .applicationId(appId)
            .build();

        DeleteAppResponse result = pinpoint.deleteApp(appRequest);
        String appName = result.applicationResponse().name();
        System.out.println("Application " + appName + " has been deleted.");

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [DeleteApp](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen eines Endpunkts

Das folgende Codebeispiel zeigt, wie Sie einen Endpunkt löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Löschen eines Endpunktes

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteEndpoint {
    public static void main(String[] args) {
        final String usage = ""

            Usage:  <appName> <endpointId >

            Where:
                appId - The id of the application to delete.
                endpointId - The id of the endpoint to delete.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String endpointId = args[1];
        System.out.println("Deleting an endpoint with id: " + endpointId);
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        deletePinEndpoint(pinpoint, appId, endpointId);
        pinpoint.close();
    }

    public static void deletePinEndpoint(PinpointClient pinpoint, String appId,
        String endpointId) {
```

```
try {
    DeleteEndpointRequest appRequest = DeleteEndpointRequest.builder()
        .applicationId(appId)
        .endpointId(endpointId)
        .build();

    DeleteEndpointResponse result = pinpoint.deleteEndpoint(appRequest);
    String id = result.endpointResponse().id();
    System.out.println("The deleted endpoint id " + id);

} catch (PinpointException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.println("Done");
}
```

- Weitere API-Informationen finden Sie unter [DeleteEndpoint](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Exportieren eines Endpunkts

Das folgende Codebeispiel zeigt, wie Sie einen Endpunkt exportieren.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Exportieren Sie einen Endpunkt.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.ExportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.CreateExportJobRequest;
```

```
import software.amazon.awssdk.services.pinpoint.model.CreateExportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.GetExportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.GetExportJobRequest;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.concurrent.TimeUnit;
import java.util.stream.Collectors;

/**
 * To run this code example, you need to create an AWS Identity and Access
 * Management (IAM) role with the correct policy as described in this
 * documentation:
 * https://docs.aws.amazon.com/pinpoint/latest/developerguide/audience-data-export.html
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ExportEndpoints {
    public static void main(String[] args) {
        final String usage = ""

                This program performs the following steps:

                1. Exports the endpoints to an Amazon S3 bucket.
                2. Downloads the exported endpoints files from Amazon S3.
    }
}
```

3. Parses the endpoints files to obtain the endpoint IDs and prints them.

```
Usage: ExportEndpoints <applicationId> <s3BucketName>
<iamExportRoleArn> <path>
```

Where:

applicationId - The ID of the Amazon Pinpoint application that has the endpoint.

s3BucketName - The name of the Amazon S3 bucket to export the JSON file to.\s

iamExportRoleArn - The ARN of an IAM role that grants Amazon Pinpoint write permissions to the S3 bucket. path - The path where the files downloaded from the Amazon S3 bucket are written (for example, C:/AWS/).

```
""";
```

```
if (args.length != 4) {
    System.out.println(usage);
    System.exit(1);
}

String applicationId = args[0];
String s3BucketName = args[1];
String iamExportRoleArn = args[2];
String path = args[3];
System.out.println("Deleting an application with ID: " + applicationId);

Region region = Region.US_EAST_1;
PinpointClient pinpoint = PinpointClient.builder()
    .region(region)
    .build();

S3Client s3Client = S3Client.builder()
    .region(region)
    .build();

exportAllEndpoints(pinpoint, s3Client, applicationId, s3BucketName, path,
iamExportRoleArn);
pinpoint.close();
s3Client.close();
}

public static void exportAllEndpoints(PinpointClient pinpoint,
    S3Client s3Client,
    String applicationId,
```

```

        String s3BucketName,
        String path,
        String iamExportRoleArn) {

    try {
        List<String> objectKeys = exportEndpointsToS3(pinpoint, s3Client,
s3BucketName, iamExportRoleArn,
            applicationId);
        List<String> endpointFileKeys = objectKeys.stream().filter(o ->
o.endsWith(".gz"))
            .collect(Collectors.toList());
        downloadFromS3(s3Client, path, s3BucketName, endpointFileKeys);

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static List<String> exportEndpointsToS3(PinpointClient pinpoint, S3Client
s3Client, String s3BucketName,
        String iamExportRoleArn, String applicationId) {

    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd-
HH_mm:ss.SSS_z");
    String endpointsKeyPrefix = "exports/" + applicationId + "_" +
dateFormat.format(new Date());
    String s3UrlPrefix = "s3://" + s3BucketName + "/" + endpointsKeyPrefix +
"/";

    List<String> objectKeys = new ArrayList<>();
    String key;

    try {
        // Defines the export job that Amazon Pinpoint runs.
        ExportJobRequest jobRequest = ExportJobRequest.builder()
            .roleArn(iamExportRoleArn)
            .s3UrlPrefix(s3UrlPrefix)
            .build();

        CreateExportJobRequest exportJobRequest =
CreateExportJobRequest.builder()
            .applicationId(applicationId)
            .exportJobRequest(jobRequest)
            .build();

```

```
        System.out.format("Exporting endpoints from Amazon Pinpoint application
%s to Amazon S3 " +
        "bucket %s . . .\n", applicationId, s3BucketName);

        CreateExportJobResponse exportResult =
pinpoint.createExportJob(exportJobRequest);
        String jobId = exportResult.exportJobResponse().id();
        System.out.println(jobId);
        printExportJobStatus(pinpoint, applicationId, jobId);

        ListObjectsV2Request v2Request = ListObjectsV2Request.builder()
            .bucket(s3BucketName)
            .prefix(endpointsKeyPrefix)
            .build();

        // Create a list of object keys.
        ListObjectsV2Response v2Response = s3Client.listObjectsV2(v2Request);
        List<S3Object> objects = v2Response.contents();
        for (S3Object object : objects) {
            key = object.key();
            objectKeys.add(key);
        }

        return objectKeys;

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

private static void printExportJobStatus(PinpointClient pinpointClient,
    String applicationId,
    String jobId) {

    GetExportJobResponse getExportJobResult;
    String status;

    try {
        // Checks the job status until the job completes or fails.
        GetExportJobRequest exportJobRequest = GetExportJobRequest.builder()
            .jobId(jobId)
```



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

    do {
        getExportJobResult = pinpointClient.getExportJob(exportJobRequest);
        status =
getExportJobResult.exportJobResponse().jobStatus().toString().toUpperCase();
        System.out.format("Export job %s . . .\n", status);
        TimeUnit.SECONDS.sleep(3);

    } while (!status.equals("COMPLETED") && !status.equals("FAILED"));

    if (status.equals("COMPLETED")) {
        System.out.println("Finished exporting endpoints.");
    } else {
        System.err.println("Failed to export endpoints.");
        System.exit(1);
    }

} catch (PinpointException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

// Download files from an Amazon S3 bucket and write them to the path location.
public static void downloadFromS3(S3Client s3Client, String path, String
s3BucketName, List<String> objectKeys) {

    String newPath;
    try {
        for (String key : objectKeys) {
            GetObjectRequest objectRequest = GetObjectRequest.builder()
                .bucket(s3BucketName)
                .key(key)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            String fileSuffix = new
SimpleDateFormat("yyyyMMddHHmmss").format(new Date());
```

```

        newPath = path + fileSuffix + ".gz";
        File myFile = new File(newPath);
        OutputStream os = new FileOutputStream(myFile);
        os.write(data);
    }
    System.out.println("Download finished.");

} catch (S3Exception | NullPointerException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
}
}

```

- Weitere API-Informationen finden Sie unter [CreateExportJob](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Endpunkte finden

Das folgende Codebeispiel zeigt, wie Endpunkte abgerufen werden.

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import com.google.gson.FieldNamingPolicy;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointRequest;

/**
 * Before running this Java V2 code example, set up your development

```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class LookUpEndpoint {
    public static void main(String[] args) {
        final String usage = ""

            Usage:  <appId> <endpoint>

            Where:
                appId - The ID of the application to delete.
                endpoint - The ID of the endpoint.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String endpoint = args[1];
        System.out.println("Looking up an endpoint point with ID: " + endpoint);
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        lookupPinpointEndpoint(pinpoint, appId, endpoint);
        pinpoint.close();
    }

    public static void lookupPinpointEndpoint(PinpointClient pinpoint, String appId,
String endpoint) {
        try {
            GetEndpointRequest appRequest = GetEndpointRequest.builder()
                .applicationId(appId)
                .endpointId(endpoint)
                .build();

            GetEndpointResponse result = pinpoint.getEndpoint(appRequest);
            EndpointResponse endResponse = result.endpointResponse();
        }
    }
}
```

```
// Uses the Google Gson library to pretty print the endpoint JSON.
Gson gson = new GsonBuilder()
    .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
    .setPrettyPrinting()
    .create();

String endpointJson = gson.toJson(endResponse);
System.out.println(endpointJson);

} catch (PinpointException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.println("Done");
}
```

- Weitere API-Informationen finden Sie unter [GetEndpoint](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Ein Segment importieren

Der folgende Beispielcode zeigt, wie Sie ein Segment importieren.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Importieren Sie ein Segment.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.ImportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.ImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.Format;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobResponse;
```

```
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ImportSegment {
    public static void main(String[] args) {
        final String usage = ""

            Usage:  <appId> <bucket> <key> <roleArn>\s

            Where:
                appId - The application ID to create a segment for.
                bucket - The name of the Amazon S3 bucket that contains the
segment definitons.
                key - The key of the S3 object.
                roleArn - ARN of the role that allows Amazon Pinpoint to
access S3. You need to set trust management for this to work. See https://docs.aws.amazon.com/IAM/latest/UserGuide/reference\_policies\_elements\_principal.html
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String bucket = args[1];
        String key = args[2];
        String roleArn = args[3];

        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        ImportJobResponse response = createImportSegment(pinpoint, appId, bucket,
key, roleArn);
        System.out.println("Import job for " + bucket + " submitted.");
    }
}
```

```
        System.out.println("See application " + response.applicationId() + " for
import job status.");
        System.out.println("See application " + response.jobStatus() + " for import
job status.");
        pinpoint.close();
    }

    public static ImportJobResponse createImportSegment(PinpointClient client,
        String appId,
        String bucket,
        String key,
        String roleArn) {

        try {
            ImportJobRequest importRequest = ImportJobRequest.builder()
                .defineSegment(true)
                .registerEndpoints(true)
                .roleArn(roleArn)
                .format(Format.JSON)
                .s3Url("s3://" + bucket + "/" + key)
                .build();

            CreateImportJobRequest jobRequest = CreateImportJobRequest.builder()
                .importJobRequest(importRequest)
                .applicationId(appId)
                .build();

            CreateImportJobResponse jobResponse =
client.createImportJob(jobRequest);
            return jobResponse.importJobResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- Weitere API-Informationen finden Sie unter [CreateImportJob](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Endpunkte auflisten

Der folgende Code zeigt, wie Sie Endpunkte auflisten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetUserEndpointsRequest;
import software.amazon.awssdk.services.pinpoint.model.GetUserEndpointsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListEndpointIds {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <applicationId> <userId>

                Where:
                applicationId - The ID of the Amazon Pinpoint application that
                has the endpoint.
                userId - The user id applicable to the endpoints""";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String applicationId = args[0];
String userId = args[1];
PinpointClient pinpoint = PinpointClient.builder()
    .region(Region.US_EAST_1)
    .build();

listAllEndpoints(pinpoint, applicationId, userId);
pinpoint.close();
}

public static void listAllEndpoints(PinpointClient pinpoint,
    String applicationId,
    String userId) {

    try {
        GetUserEndpointsRequest endpointsRequest =
        GetUserEndpointsRequest.builder()
            .userId(userId)
            .applicationId(applicationId)
            .build();

        GetUserEndpointsResponse response =
        pinpoint.getUserEndpoints(endpointsRequest);
        List<EndpointResponse> endpoints = response.endpointsResponse().item();

        // Display the results.
        for (EndpointResponse endpoint : endpoints) {
            System.out.println("The channel type is: " +
            endpoint.channelType());
            System.out.println("The address is " + endpoint.address());
        }

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [GetUserEndpoints](#) in der APIAWS SDK for Java 2.x -Referenz für .



## Segmente auflisten

Das folgende Codebeispiel zeigt, wie Segmente aufgelistet werden.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Listen Sie Segmente auf.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsRequest;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListSegments {
    public static void main(String[] args) {
        final String usage = ""

            Usage:  <appId>

            Where:
                appId - The ID of the application that contains a segment.

            """;

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```

```

        System.exit(1);
    }

    String appId = args[0];
    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

    listSegs(pinpoint, appId);
    pinpoint.close();
}

public static void listSegs(PinpointClient pinpoint, String appId) {
    try {
        GetSegmentsRequest request = GetSegmentsRequest.builder()
            .applicationId(appId)
            .build();

        GetSegmentsResponse response = pinpoint.getSegments(request);
        List<SegmentResponse> segments = response.segmentsResponse().item();
        for (SegmentResponse segment : segments) {
            System.out
                .println("Segment " + segment.id() + " " + segment.name() +
                    " " + segment.lastModifiedDate());
        }

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Weitere API-Informationen finden Sie unter [GetSegments](#) in der APIAWS SDK for Java 2.x - Referenz für .

## E-Mails und Textnachrichten senden

Das folgende Codebeispiel zeigt, wie E-Mail- und Textnachrichten mit Amazon Pinpoint gesendet werden.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Senden Sie eine E-Mail-Nachricht.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.SimpleEmailPart;
import software.amazon.awssdk.services.pinpoint.model.SimpleEmail;
import software.amazon.awssdk.services.pinpoint.model.EmailMessage;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;
import software.amazon.awssdk.services.pinpointemail.model.Body;
import software.amazon.awssdk.services.pinpointemail.model.Content;
import software.amazon.awssdk.services.pinpointemail.model.Destination;
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;
import software.amazon.awssdk.services.pinpointemail.model.Message;
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;

import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendEmailMessage {

    // The character encoding the you want to use for the subject line and
```

```
// message body of the email.
public static String charset = "UTF-8";

// The body of the email for recipients whose email clients support HTML
content.
static final String body = ""
    Amazon Pinpoint test (AWS SDK for Java 2.x)

    This email was sent through the Amazon Pinpoint Email API using the AWS SDK
for Java 2.x

    """;

public static void main(String[] args) {
    final String usage = ""

        Usage:    <subject> <appId> <senderAddress>
<toAddress>

        Where:
            subject - The email subject to use.
            senderAddress - The from address. This address has to be verified in
Amazon Pinpoint in the region you're using to send email\s
            toAddress - The to address. This address has to be verified in Amazon
Pinpoint in the region you're using to send email\s
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String subject = args[0];
    String senderAddress = args[1];
    String toAddress = args[2];
    System.out.println("Sending a message");
    PinpointEmailClient pinpoint = PinpointEmailClient.builder()
        .region(Region.US_EAST_1)
        .build();

    sendEmail(pinpoint, subject, senderAddress, toAddress);
    System.out.println("Email was sent");
    pinpoint.close();
}
```

```
public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
subject, String senderAddress, String toAddress) {
    try {
        Content content = Content.builder()
            .data(body)
            .build();

        Body messageBody = Body.builder()
            .text(content)
            .build();

        Message message = Message.builder()
            .body(messageBody)
            .subject(Content.builder().data(subject).build())
            .build();

        Destination destination = Destination.builder()
            .toAddresses(toAddress)
            .build();

        EmailContent emailContent = EmailContent.builder()
            .simple(message)
            .build();

        SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
            .fromEmailAddress(senderAddress)
            .destination(destination)
            .content(emailContent)
            .build();

        pinpointEmailClient.sendEmail(sendEmailRequest);
        System.out.println("Message Sent");

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Senden einer E-Mail-Nachricht mit CC-Werten.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;
import software.amazon.awssdk.services.pinpointemail.model.Body;
import software.amazon.awssdk.services.pinpointemail.model.Content;
import software.amazon.awssdk.services.pinpointemail.model.Destination;
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;
import software.amazon.awssdk.services.pinpointemail.model.Message;
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendEmailMessageCC {

    // The body of the email.
    static final String body = ""
        Amazon Pinpoint test (AWS SDK for Java 2.x)

        This email was sent through the Amazon Pinpoint Email API using the AWS SDK
    for Java 2.x

    "";
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <subject> <senderAddress> <toAddress> <ccAddress>

            Where:
                subject - The email subject to use.
                senderAddress - The from address. This address has to be verified in
Amazon Pinpoint in the region you're using to send email\s
                toAddress - The to address. This address has to be verified in Amazon
Pinpoint in the region you're using to send email\s
                ccAddress - The CC address.

            "";
```

```
    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String subject = args[0];
    String senderAddress = args[1];
    String toAddress = args[2];
    String ccAddress = args[3];

    System.out.println("Sending a message");
    PinpointEmailClient pinpoint = PinpointEmailClient.builder()
        .region(Region.US_EAST_1)
        .build();

    ArrayList<String> ccList = new ArrayList<>();
    ccList.add(ccAddress);
    sendEmail(pinpoint, subject, senderAddress, toAddress, ccList);
    pinpoint.close();
}

public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
subject, String senderAddress, String toAddress, ArrayList<String> ccAddresses) {
    try {
        Content content = Content.builder()
            .data(body)
            .build();

        Body messageBody = Body.builder()
            .text(content)
            .build();

        Message message = Message.builder()
            .body(messageBody)
            .subject(Content.builder().data(subject).build())
            .build();

        Destination destination = Destination.builder()
            .toAddresses(toAddress)
            .ccAddresses(ccAddresses)
            .build();

        EmailContent emailContent = EmailContent.builder()
            .simple(message)
```

```
        .build();

        SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
            .fromEmailAddress(senderAddress)
            .destination(destination)
            .content(emailContent)
            .build();

        pinpointEmailClient.sendEmail(sendEmailRequest);
        System.out.println("Message Sent");

    } catch (PinpointException e) {
        // Handle exception
        e.printStackTrace();
    }
}
}
```

Senden Sie eine SMS-Nachricht.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessageResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```



```

public class SendMessage {

    // The type of SMS message that you want to send. If you plan to send
    // time-sensitive content, specify TRANSACTIONAL. If you plan to send
    // marketing-related content, specify PROMOTIONAL.
    public static String messageType = "TRANSACTIONAL";

    // The registered keyword associated with the originating short code.
    public static String registeredKeyword = "myKeyword";

    // The sender ID to use when sending the message. Support for sender ID
    // varies by country or region. For more information, see
    // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
    countries.html
    public static String senderId = "MySenderId";

    public static void main(String[] args) {
        final String usage = ""

            Usage:  <message> <appId> <originationNumber>
<destinationNumber>\s

            Where:
                message - The body of the message to send.
                appId - The Amazon Pinpoint project/application ID
to use when you send this message.
                originationNumber - The phone number or short code
that you specify has to be associated with your Amazon Pinpoint account. For best
results, specify long codes in E.164 format (for example, +1-555-555-5654).
                destinationNumber - The recipient's phone number.
For best results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).\s

            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String appId = args[1];
        String originationNumber = args[2];
        String destinationNumber = args[3];
        System.out.println("Sending a message");
    }
}

```

```
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber);
        pinpoint.close();
    }

    public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId,
        String originationNumber,
        String destinationNumber) {
        try {
            Map<String, AddressConfiguration> addressMap = new
HashMap<String, AddressConfiguration>();
            AddressConfiguration addConfig =
AddressConfiguration.builder()
                .channelType(ChannelType.SMS)
                .build();

            addressMap.put(destinationNumber, addConfig);
            SMSMessage smsMessage = SMSMessage.builder()
                .body(message)
                .messageType(messageType)
                .originationNumber(originationNumber)
                .senderId(senderId)
                .keyword(registeredKeyword)
                .build();

            // Create a DirectMessageConfiguration object.
            DirectMessageConfiguration direct =
DirectMessageConfiguration.builder()
                .smsMessage(smsMessage)
                .build();

            MessageRequest msgReq = MessageRequest.builder()
                .addresses(addressMap)
                .messageConfiguration(direct)
                .build();

            // create a SendMessagesRequest object
            SendMessagesRequest request = SendMessagesRequest.builder()
                .applicationId(appId)
```

```

                .messageRequest(msgReq)
                .build();

        SendMessagesResponse response =
pinpoint.sendMessage(request);
        MessageResponse msg1 = response.messageResponse();
        Map map1 = msg1.result();

        // Write out the result of sendMessage.
        map1.forEach((k, v) -> System.out.println((k + ":" + v)));

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

## Senden Sie Batch-SMS-Nachrichten.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

```

```
public class SendMessageBatch {

    // The type of SMS message that you want to send. If you plan to send
    // time-sensitive content, specify TRANSACTIONAL. If you plan to send
    // marketing-related content, specify PROMOTIONAL.
    public static String messageType = "TRANSACTIONAL";

    // The registered keyword associated with the originating short code.
    public static String registeredKeyword = "myKeyword";

    // The sender ID to use when sending the message. Support for sender ID
    // varies by country or region. For more information, see
    // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-countries.html
    public static String senderId = "MySenderId";

    public static void main(String[] args) {
        final String usage = ""

            Usage:  <message> <appId> <originationNumber>
<destinationNumber> <destinationNumber1>\s

            Where:
                message - The body of the message to send.
                appId - The Amazon Pinpoint project/application ID
to use when you send this message.
                originationNumber - The phone number or short code
that you specify has to be associated with your Amazon Pinpoint account. For best
results, specify long codes in E.164 format (for example, +1-555-555-5654).
                destinationNumber - The recipient's phone number.
For best results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).
                destinationNumber1 - The second recipient's phone
number. For best results, you should specify the phone number in E.164 format (for
example, +1-555-555-5654).\s

            """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String appId = args[1];
```

```
String originationNumber = args[2];
String destinationNumber = args[3];
String destinationNumber1 = args[4];
System.out.println("Sending a message");
PinpointClient pinpoint = PinpointClient.builder()
    .region(Region.US_EAST_1)
    .build();

    sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber, destinationNumber1);
    pinpoint.close();
}

public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId,
    String originationNumber,
    String destinationNumber, String destinationNumber1) {
    try {
        Map<String, AddressConfiguration> addressMap = new
HashMap<String, AddressConfiguration>();
        AddressConfiguration addConfig =
AddressConfiguration.builder()
            .channelType(ChannelType.SMS)
            .build();

        // Add an entry to the Map object for each number to whom
you want to send a
        // message.
        addressMap.put(destinationNumber, addConfig);
        addressMap.put(destinationNumber1, addConfig);
        SMSMessage smsMessage = SMSMessage.builder()
            .body(message)
            .messageType(messageType)
            .originationNumber(originationNumber)
            .senderId(senderId)
            .keyword(registeredKeyword)
            .build();

        // Create a DirectMessageConfiguration object.
        DirectMessageConfiguration direct =
DirectMessageConfiguration.builder()
            .smsMessage(smsMessage)
            .build();
```

```
        MessageRequest msgReq = MessageRequest.builder()
            .addresses(addressMap)
            .messageConfiguration(direct)
            .build();

        // Create a SendMessagesRequest object.
        SendMessagesRequest request = SendMessagesRequest.builder()
            .applicationId(appId)
            .messageRequest(msgReq)
            .build();

        SendMessagesResponse response =
pinpoint.sendMessage(request);
        MessageResponse msg1 = response.messageResponse();
        Map map1 = msg1.result();

        // Write out the result of sendMessage.
        map1.forEach((k, v) -> System.out.println((k + ":" + v)));

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [SendMessages](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Einen Endpunkt aktualisieren

Der folgende Code zeigt, wie Sie einen Endpunkt aktualisieren.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.EndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.EndpointDemographic;
import software.amazon.awssdk.services.pinpoint.model.EndpointLocation;
import software.amazon.awssdk.services.pinpoint.model.EndpointUser;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.List;
import java.util.UUID;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Date;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateEndpoint {
    public static void main(String[] args) {
        final String usage = ""

                Usage: <appId>

                Where:
                    appId - The ID of the application to create an endpoint for.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String appId = args[0];
    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

    EndpointResponse response = createEndpoint(pinpoint, appId);
    System.out.println("Got Endpoint: " + response.id());
    pinpoint.close();
}

public static EndpointResponse createEndpoint(PinpointClient client, String
appId) {
    String endpointId = UUID.randomUUID().toString();
    System.out.println("Endpoint ID: " + endpointId);

    try {
        EndpointRequest endpointRequest = createEndpointRequestData();
        UpdateEndpointRequest updateEndpointRequest =
UpdateEndpointRequest.builder()
            .applicationId(appId)
            .endpointId(endpointId)
            .endpointRequest(endpointRequest)
            .build();

        UpdateEndpointResponse updateEndpointResponse =
client.updateEndpoint(updateEndpointRequest);
        System.out.println("Update Endpoint Response: " +
updateEndpointResponse.messageBody());

        GetEndpointRequest getEndpointRequest = GetEndpointRequest.builder()
            .applicationId(appId)
            .endpointId(endpointId)
            .build();

        GetEndpointResponse getEndpointResponse =
client.getEndpoint(getEndpointRequest);
        System.out.println(getEndpointResponse.endpointResponse().address());

        System.out.println(getEndpointResponse.endpointResponse().channelType());

        System.out.println(getEndpointResponse.endpointResponse().applicationId());
    }
}
```



```
System.out.println(getEndpointResponse.endpointResponse().endpointStatus());
    System.out.println(getEndpointResponse.endpointResponse().requestId());
    System.out.println(getEndpointResponse.endpointResponse().user());

    return getEndpointResponse.endpointResponse();

} catch (PinpointException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
}

private static EndpointRequest createEndpointRequestData() {
    try {
        List<String> favoriteTeams = new ArrayList<>();
        favoriteTeams.add("Lakers");
        favoriteTeams.add("Warriors");
        HashMap<String, List<String>> customAttributes = new HashMap<>();
        customAttributes.put("team", favoriteTeams);

        EndpointDemographic demographic = EndpointDemographic.builder()
            .appVersion("1.0")
            .make("apple")
            .model("iPhone")
            .modelVersion("7")
            .platform("ios")
            .platformVersion("10.1.1")
            .timezone("America/Los_Angeles")
            .build();

        EndpointLocation location = EndpointLocation.builder()
            .city("Los Angeles")
            .country("US")
            .latitude(34.0)
            .longitude(-118.2)
            .postalCode("90068")
            .region("CA")
            .build();

        Map<String, Double> metrics = new HashMap<>();
        metrics.put("health", 100.00);
        metrics.put("luck", 75.00);
    }
}
```

```

        EndpointUser user = EndpointUser.builder()
            .userId(UUID.randomUUID().toString())
            .build();

        DateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm'Z'"); // Quoted
        "Z" to indicate UTC, no timezone                                     // offset

        String nowAsISO = df.format(new Date());

        return EndpointRequest.builder()
            .address(UUID.randomUUID().toString())
            .attributes(customAttributes)
            .channelType("APNS")
            .demographic(demographic)
            .effectiveDate(nowAsISO)
            .location(location)
            .metrics(metrics)
            .optOut("NONE")
            .requestId(UUID.randomUUID().toString())
            .user(user)
            .build();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
}

```

- Weitere API-Informationen finden Sie unter [UpdateEndpoint](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Kanäle aktualisieren

Das folgende Codebeispiel zeigt, wie Sie Kanäle aktualisieren.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.SMSChannelResponse;
import software.amazon.awssdk.services.pinpoint.model.GetSmsChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SMSChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateSmsChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateSmsChannelResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateChannel {
    public static void main(String[] args) {
        final String usage = ""

                Usage: CreateChannel <appId>

                Where:
                appId - The name of the application whose channel is updated.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
```

```
PinpointClient pinpoint = PinpointClient.builder()
    .region(Region.US_EAST_1)
    .build();

SMSChannelResponse getResponse = getSmsChannel(pinpoint, appId);
toggleSmsChannel(pinpoint, appId, getResponse);
pinpoint.close();
}

private static SMSChannelResponse getSmsChannel(PinpointClient client, String
appId) {
    try {
        GetSmsChannelRequest request = GetSmsChannelRequest.builder()
            .applicationId(appId)
            .build();

        SMSChannelResponse response =
client.getSmsChannel(request).smsChannelResponse();
        System.out.println("Channel state is " + response.enabled());
        return response;

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

private static void toggleSmsChannel(PinpointClient client, String appId,
SMSChannelResponse getResponse) {
    boolean enabled = !getResponse.enabled();
    try {
        SMSChannelRequest request = SMSChannelRequest.builder()
            .enabled(enabled)
            .build();

        UpdateSmsChannelRequest updateRequest =
UpdateSmsChannelRequest.builder()
            .smsChannelRequest(request)
            .applicationId(appId)
            .build();

        UpdateSmsChannelResponse result =
client.updateSmsChannel(updateRequest);
    }
}
```

```
        System.out.println("Channel state: " +
result.smsChannelResponse().enabled());

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [GetSmsChannel](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Amazon Pinpoint-SMS- und Sprachnachrichten-API-Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit der Amazon Pinpoint-SMS- und Sprachnachrichten-API verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)

### Aktionen

Senden Sie eine Sprachnachricht mit Amazon-Pinpoint-SMS- und -Sprachnachrichten-API

Das folgende Codebeispiel zeigt, wie eine Sprachnachricht mit der Amazon Pinpoint-SMS- und Sprachnachrichten-API gesendet wird.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpointsmsvoice.PinpointSmsVoiceClient;
import software.amazon.awssdk.services.pinpointsmsvoice.model.SSMLMessageType;
import software.amazon.awssdk.services.pinpointsmsvoice.model.VoiceMessageContent;
import
    software.amazon.awssdk.services.pinpointsmsvoice.model.SendVoiceMessageRequest;
import
    software.amazon.awssdk.services.pinpointsmsvoice.model.PinpointSmsVoiceException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendVoiceMessage {

    // The Amazon Polly voice that you want to use to send the message. For a
    list
    // of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html
    static final String voiceName = "Matthew";

    // The language to use when sending the message. For a list of supported
    // languages, see
    // https://docs.aws.amazon.com/polly/latest/dg/SupportedLanguage.html
    static final String languageCode = "en-US";
```

```

// The content of the message. This example uses SSML to customize and
control
// certain aspects of the message, such as by adding pauses and changing
// phonation. The message can't contain any line breaks.
static final String ssmlMessage = "<speak>This is a test message sent from "
    + "<emphasis>Amazon Pinpoint</emphasis> "
    + "using the <break strength='weak'/>AWS "
    + "SDK for Java. "
    + "<amazon:effect phonation='soft'>Thank "
    + "you for listening.</amazon:effect></speak>";

public static void main(String[] args) {

    final String usage = ""

        Usage:  <originationNumber> <destinationNumber>\s

        Where:
            originationNumber - The phone number or short code
that you specify has to be associated with your Amazon Pinpoint account. For best
results, specify long codes in E.164 format (for example, +1-555-555-5654).
            destinationNumber - The recipient's phone number.
For best results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).\s

        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String originationNumber = args[0];
    String destinationNumber = args[1];
    System.out.println("Sending a voice message");

    // Set the content type to application/json.
    List<String> listVal = new ArrayList<>();
    listVal.add("application/json");
    Map<String, List<String>> values = new HashMap<>();
    values.put("Content-Type", listVal);

    ClientOverrideConfiguration config2 =
ClientOverrideConfiguration.builder()
        .headers(values)

```

```
        .build();

        PinpointSmsVoiceClient client = PinpointSmsVoiceClient.builder()
            .overrideConfiguration(config2)
            .region(Region.US_EAST_1)
            .build();

        sendVoiceMsg(client, originationNumber, destinationNumber);
        client.close();
    }

    public static void sendVoiceMsg(PinpointSmsVoiceClient client, String
originationNumber,
        String destinationNumber) {
        try {
            SSMLMessageType ssmlMessageType = SSMLMessageType.builder()
                .languageCode(languageCode)
                .text(ssmlMessage)
                .voiceId(voiceName)
                .build();

            VoiceMessageContent content = VoiceMessageContent.builder()
                .ssmlMessage(ssmlMessageType)
                .build();

            SendVoiceMessageRequest voiceMessageRequest =
SendVoiceMessageRequest.builder()
                .destinationPhoneNumber(destinationNumber)
                .originationPhoneNumber(originationNumber)
                .content(content)
                .build();

            client.sendVoiceMessage(voiceMessageRequest);
            System.out.println("The message was sent successfully.");

        } catch (PinpointSmsVoiceException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```



- Weitere API-Informationen finden Sie unter [SendVoiceMessage](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Amazon Polly-Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon Polly verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)

### Aktionen

Holen Sie sich Stimmen zur Generierung

Das folgende Codebeispiel zeigt, wie Sie Amazon Polly-Stimmen für die Synthetisierung zur Verfügung stellen.

SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;
```

```
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;
import software.amazon.awssdk.services.polly.model.PollyException;
import software.amazon.awssdk.services.polly.model.Voice;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeVoicesSample {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
            .build();

        describeVoice(polly);
        polly.close();
    }

    public static void describeVoice(PollyClient polly) {
        try {
            DescribeVoicesRequest voicesRequest = DescribeVoicesRequest.builder()
                .languageCode("en-US")
                .build();

            DescribeVoicesResponse enUsVoicesResult =
polly.describeVoices(voicesRequest);
            List<Voice> voices = enUsVoicesResult.voices();
            for (Voice myVoice : voices) {
                System.out.println("The ID of the voice is " + myVoice.id());
                System.out.println("The gender of the voice is " +
myVoice.gender());
            }

        } catch (PollyException e) {
            System.err.println("Exception caught: " + e);
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeVoices](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Aussprachelexika auflisten

Das folgende Codebeispiel zeigt, wie Sie Amazon Polly-Aussprachelexika auflisten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.ListLexiconsResponse;
import software.amazon.awssdk.services.polly.model.ListLexiconsRequest;
import software.amazon.awssdk.services.polly.model.LexiconDescription;
import software.amazon.awssdk.services.polly.model.PollyException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListLexicons {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
            .build();

        listLexicons(polly);
        polly.close();
    }
}
```

```
public static void listLexicons(PollyClient client) {
    try {
        ListLexiconsRequest listLexiconsRequest = ListLexiconsRequest.builder()
            .build();

        ListLexiconsResponse listLexiconsResult =
client.listLexicons(listLexiconsRequest);
        List<LexiconDescription> lexiconDescription =
listLexiconsResult.lexicons();
        for (LexiconDescription lexDescription : lexiconDescription) {
            System.out.println("The name of the Lexicon is " +
lexDescription.name());
        }

    } catch (PollyException e) {
        System.err.println("Exception caught: " + e);
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [ListLexicons](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Synthetisieren von Sprache aus Text

Das folgende Codebeispiel zeigt, wie Sie Sprache aus Text mit Amazon Polly synthetisieren.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import javazoom.jl.decoder.JavaLayerException;
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
```

```
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;
import software.amazon.awssdk.services.polly.model.Voice;
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;
import software.amazon.awssdk.services.polly.model.OutputFormat;
import software.amazon.awssdk.services.polly.model.PollyException;
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechRequest;
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechResponse;
import java.io.IOException;
import java.io.InputStream;
import javazoom.jl.player.advanced.AdvancedPlayer;
import javazoom.jl.player.advanced.PlaybackEvent;
import javazoom.jl.player.advanced.PlaybackListener;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PollyDemo {
    private static final String SAMPLE = "Congratulations. You have successfully
    built this working demo " +
        " of Amazon Polly in Java Version 2. Have fun building voice enabled
    apps with Amazon Polly (that's me!), and always "
        +
        " look at the AWS website for tips and tricks on using Amazon Polly and
    other great services from AWS";

    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
            .build();

        talkPolly(polly);
        polly.close();
    }

    public static void talkPolly(PollyClient polly) {
        try {
            DescribeVoicesRequest describeVoiceRequest =
DescribeVoicesRequest.builder()
                .engine("standard")
```

```

        .build());

        DescribeVoicesResponse describeVoicesResult =
polly.describeVoices(describeVoiceRequest);
        Voice voice = describeVoicesResult.voices().stream()
            .filter(v -> v.name().equals("Joanna"))
            .findFirst()
            .orElseThrow(() -> new RuntimeException("Voice not found"));
        InputStream stream = synthesize(polly, SAMPLE, voice, OutputFormat.MP3);
        AdvancedPlayer player = new AdvancedPlayer(stream,

javazoom.jl.player.FactoryRegistry.systemRegistry().createAudioDevice());
        player.setPlaybackListener(new PlaybackListener() {
            public void playbackStarted(PlaybackEvent evt) {
                System.out.println("Playback started");
                System.out.println(SAMPLE);
            }

            public void playbackFinished(PlaybackEvent evt) {
                System.out.println("Playback finished");
            }
        });

        // play it!
        player.play();

    } catch (PollyException | JavaLayerException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static InputStream synthesize(PollyClient polly, String text, Voice
voice, OutputFormat format)
    throws IOException {
    SynthesizeSpeechRequest synthReq = SynthesizeSpeechRequest.builder()
        .text(text)
        .voiceId(voice.id())
        .outputFormat(format)
        .build();

    ResponseInputStream<SynthesizeSpeechResponse> synthRes =
polly.synthesizeSpeech(synthReq);
    return synthRes;
}

```

```
}  
}
```

- Weitere API-Informationen finden Sie unter [SynthesizeSpeech](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Amazon-RDS-Beispiele mit SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon RDS verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Erste Schritte

#### Hello Amazon RDS

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon RDS.

#### SDK für Java 2.x

##### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.rds.RdsClient;  
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;  
import software.amazon.awssdk.services.rds.model.DBInstance;  
import software.amazon.awssdk.services.rds.model.RdsException;
```

```
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }

    public static void describeInstances(RdsClient rdsClient) {
        try {
            DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                System.out.println("Instance ARN is: " + instance.dbInstanceArn());
                System.out.println("The Engine is " + instance.engine());
                System.out.println("Connection endpoint is" +
instance.endpoint().address());
            }

            } catch (RdsException e) {
                System.out.println(e.getLocalizedMessage());
                System.exit(1);
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeDBInstances](#) in der API-Referenz zu AWS SDK for Java 2.x .



## Themen

- [Aktionen](#)
- [Szenarien](#)

## Aktionen

Erstellen einer DB-Instance.

Das folgende Codebeispiel zeigt, wie Sie eine Amazon RDS-DB-Instance erstellen und warten, bis sie verfügbar ist.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import com.google.gson.Gson;
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* This example requires an AWS Secrets Manager secret that contains the
* database credentials. If you do not create a
* secret, this example will not work. For more details, see:
*
* https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating\_how-
services-use-secrets\_RS.html
*
*/

public class CreateDBInstance {
    public static long sleepTime = 20;

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <dbInstanceIdentifier> <dbName> <secretName>

            Where:
                dbInstanceIdentifier - The database instance identifier.\s
                dbName - The database name.\s
                secretName - The name of the AWS Secrets Manager secret that
contains the database credentials."
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String dbName = args[1];
        String secretName = args[2];
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();
```

```
        createDatabaseInstance(rdsClient, dbInstanceIdentifier, dbName,
user.getUsername(), user.getPassword());
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
            .region(region)

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

    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
            .secretId(secretName)
            .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    public static void createDatabaseInstance(RdsClient rdsClient,
        String dbInstanceIdentifier,
        String dbName,
        String userName,
        String userPassword) {

        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .allocatedStorage(100)
                .dbName(dbName)
                .engine("mysql")
                .dbInstanceClass("db.m4.large")
                .engineVersion("8.0")
                .storageType("standard")
                .masterUsername(userName)
```

```
        .masterUserPassword(userPassword)
        .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");
    try {
        DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .build();

        // Loop until the cluster is ready.
        while (!instanceReady) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                instanceReadyStr = instance.dbInstanceStatus();
                if (instanceReadyStr.contains("available"))
                    instanceReady = true;
                else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }
        System.out.println("Database instance is available!");
    } catch (RdsException | InterruptedException e) {
```

```
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [CreateDBInstance](#) in der AWS SDK for Java 2.x - API-Referenz.

## Erstellen einer DB-Parametergruppe

Das folgende Codebeispiel zeigt, wie Sie eine Amazon RDS-DB-Parametergruppe erstellen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
    try {
        CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
        .dbParameterGroupName(dbGroupName)
        .dbParameterGroupFamily(dbParameterGroupFamily)
        .description("Created by using the AWS SDK for Java")
        .build();

        CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());


    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [CreateDBParameterGroup](#) in der APIAWS SDK for Java 2.x -Referenz für .

So erstellen Sie einen Snapshot einer DB-Instance

Das folgende Codebeispiel zeigt, wie Sie einen Snapshot einer Amazon RDS-DB-Instance erstellen.

SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
// Create an Amazon RDS snapshot.
public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
    try {
        CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .dbSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
        System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [CreateDBSnapshot](#) in der AWS SDK for Java 2.x - API-Referenz.

## Erstellen eines Authentifizierungstokens

Das folgende Codebeispiel veranschaulicht, wie Sie ein Authentifizierungstoken für die IAM-Authentifizierung erstellen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Verwenden Sie die [RdsUtilities](#) Klasse , um ein Authentifizierungstoken zu generieren.

```
public class GenerateRDSAuthToken {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <dbInstanceIdentifier> <masterUsername>

                Where:
                dbInstanceIdentifier - The database instance identifier.\s
                masterUsername - The master user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String masterUsername = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        String token = getAuthToken(rdsClient, dbInstanceIdentifier,
masterUsername);
        System.out.println("The token response is " + token);
    }
}
```

```
public static String getAuthToken(RdsClient rdsClient, String
dbInstanceIdentifier, String masterUsername) {

    RdsUtilities utilities = rdsClient.utilities();
    try {
        GenerateAuthenticationTokenRequest tokenRequest =
GenerateAuthenticationTokenRequest.builder()
        .credentialsProvider(ProfileCredentialsProvider.create())
        .username(masterUsername)
        .port(3306)
        .hostname(dbInstanceIdentifier)
        .build();

        return utilities.generateAuthenticationToken(tokenRequest);

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Weitere API-Informationen finden Sie unter [GenerateRDSAuthToken](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Löschen einer DB-Instance

Das folgende Codebeispiel zeigt, wie Sie eine Amazon RDS-DB-Instance löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
```



```
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDBInstance {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <dbInstanceIdentifier>\s

            Where:
                dbInstanceIdentifier - The database instance identifier\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
```

```

        .deleteAutomatedBackups(true)
        .skipFinalSnapshot(true)
        .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.print("The status of the database is " +
response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}

```

- Weitere API-Informationen finden Sie unter [DeleteDBInstance](#) in der API-Referenz zu AWS SDK for Java 2.x .

## Löschen einer Parametergruppe

Das folgende Codebeispiel zeigt, wie Sie eine Amazon RDS-DB-Parametergruppe löschen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

// Delete the parameter group after database has been deleted.
// An exception is thrown if you attempt to delete the para group while database
// exists.
public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
    throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

```

```

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            didFind = false;
            int index = 1;
            for (DBInstance instance : instanceList) {
                instanceARN = instance.dbInstanceArn();
                if (instanceARN.compareTo(dbARN) == 0) {
                    System.out.println(dbARN + " still exists");
                    didFind = true;
                }
                if ((index == listSize) && (!didFind)) {
                    // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            // Delete the para group.
            DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .build();

            rdsClient.deleteDBParameterGroup(parameterGroupRequest);
            System.out.println(dbGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}

```

- Weitere API-Informationen finden Sie unter [DeleteDBParameterGroup](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Beschreiben von DB-Instances

Das folgende Codebeispiel zeigt, wie Sie Amazon-RDS-DB-Instances beschreiben.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }

    public static void describeInstances(RdsClient rdsClient) {
        try {
            DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
        }
    }
}
```

```
        for (DBInstance instance : instanceList) {
            System.out.println("Instance ARN is: " + instance.dbInstanceArn());
            System.out.println("The Engine is " + instance.engine());
            System.out.println("Connection endpoint is" +
instance.endpoint().address());
        }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeDBInstances](#) in der API-Referenz zu AWS SDK for Java 2.x .

## Beschreiben von DB-Parametergruppen

Das folgende Codebeispiel zeigt, wie Sie Amazon-RDS-DB-Parametergruppen beschreiben.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
    try {
        DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .maxRecords(20)
            .build();

        DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
        List<DBParameterGroup> groups = response.dbParameterGroups();
```

```
        for (DBParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbParameterGroupName());
            System.out.println("The group description is " +
group.description());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeDBParameterGroups](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Beschreiben der Datenbank-Engine-Versionen

Das folgende Codebeispiel zeigt, wie Sie Versionen der Amazon-RDS-Datenbank-Engine beschreiben.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .defaultOnly(true)
            .engine("mysql")
            .maxRecords(20)
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
    }
}
```

```
List<DBEngineVersion> engines = response.dbEngineVersions();

// Get all DBEngineVersion objects.
for (DBEngineVersion engineOb : engines) {
    System.out.println("The name of the DB parameter group family for
the database engine is "
        + engineOb.dbParameterGroupFamily());
    System.out.println("The name of the database engine " +
engineOb.engine());
    System.out.println("The version number of the database engine " +
engineOb.engineVersion());
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [DescribeDBEngineVersions](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Beschreiben der Optionen für DB-Instances

Das folgende Codebeispiel zeigt, wie Optionen für Amazon RDS-DB-Instances beschrieben werden.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
    try {
        DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .dbParameterGroupFamily(dbParameterGroupFamily)
```

```

        .engine("mysql")
        .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
        List<DBEngineVersion> dbEngines = response.dbEngineVersions();
        for (DBEngineVersion dbEngine : dbEngines) {
            System.out.println("The engine version is " +
dbEngine.engineVersion());
            System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

```

- Weitere API-Informationen finden Sie unter [DescribeOrderableDBInstanceOptions](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Beschreiben von Parametern in einer DB-Parametergruppe

Das folgende Codebeispiel zeigt, wie Parameter in einer Amazon-RDS-DB-Parametergruppe beschrieben werden.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

// Retrieve parameters in the group.
public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
int flag) {
    try {
        DescribeDbParametersRequest dbParameterGroupsRequest;

```



```

        if (flag == 0) {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .build();
        } else {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .source("user")
                .build();
        }

        DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para : dbParameters) {
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            paraName = para.parameterName();
            if ((paraName.compareTo("auto_increment_offset") == 0)
                || (paraName.compareTo("auto_increment_increment ") == 0)) {
                System.out.println("*** The parameter name is " + paraName);
                System.out.println("*** The parameter value is " +
para.parameterValue());
                System.out.println("*** The parameter data type is " +
para.dataType());
                System.out.println("*** The parameter description is " +
para.description());
                System.out.println("*** The parameter allowed values is " +
para.allowedValues());
            }
        }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}

```

- Weitere API-Informationen finden Sie unter [DescribeDBParameters](#) in der API-Referenz zu AWS SDK for Java 2.x .

## Ändern einer DB-Instance

Das folgende Codebeispiel zeigt, wie Sie eine Amazon RDS-DB-Instance ändern.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ModifyDBInstance {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <dbInstanceIdentifier> <dbSnapshotIdentifier>\s
                Where:
                dbInstanceIdentifier - The database instance identifier.\s
                masterUserPassword - The updated password that corresponds to
the master user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String dbInstanceIdentifier = args[0];
String masterUserPassword = args[1];
Region region = Region.US_WEST_2;
RdsClient rdsClient = RdsClient.builder()
    .region(region)
    .build();

updateIntance(rdsClient, dbInstanceIdentifier, masterUserPassword);
rdsClient.close();
}

public static void updateIntance(RdsClient rdsClient, String
dbInstanceIdentifier, String masterUserPassword) {
    try {
        // For a demo - modify the DB instance by modifying the master password.
        ModifyDbInstanceRequest modifyDbInstanceRequest =
ModifyDbInstanceRequest.builder()
    .dbInstanceIdentifier(dbInstanceIdentifier)
    .publiclyAccessible(true)
    .masterUserPassword(masterUserPassword)
    .build();

        ModifyDbInstanceResponse instanceResponse =
rdsClient.modifyDBInstance(modifyDbInstanceRequest);
        System.out.print("The ARN of the modified database is: " +
instanceResponse.dbInstance().dbInstanceArn());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [ModifyDBInstance](#) in der API-Referenz zu AWS SDK for Java 2.x .

## Neustarten einer DB-Instance

Die folgenden Schritte zeigen, wie Sie eine Amazon RDS-DB-Instance neu starten.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RebootDBInstance {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <dbInstanceIdentifier>\s

                Where:
                dbInstanceIdentifier - The database instance identifier\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
```

```
        .build();

        rebootInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void rebootInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
        try {
            RebootDbInstanceRequest rebootDbInstanceRequest =
RebootDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .build();

            RebootDbInstanceResponse instanceResponse =
rdsClient.rebootDBInstance(rebootDbInstanceRequest);
            System.out.print("The database " +
instanceResponse.dbInstance().dbInstanceArn() + " was rebooted");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [RebootDBInstance](#) in der API-Referenz zu AWS SDK for Java 2.x .

## Abrufen von Attributen

Das folgende Codebeispiel zeigt, wie Attribute abgerufen werden, die zu einem Amazon-RDS-Konto gehören.

### SDK für Java 2.x

#### Note

Auf [GitHub](#) gibt es mehr Beispiele. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.AccountQuota;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeAccountAttributesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAccountAttributes {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        getAccountAttributes(rdsClient);
        rdsClient.close();
    }

    public static void getAccountAttributes(RdsClient rdsClient) {
        try {
            DescribeAccountAttributesResponse response =
rdsClient.describeAccountAttributes();
            List<AccountQuota> quotasList = response.accountQuotas();
            for (AccountQuota quotas : quotasList) {
                System.out.println("Name is: " + quotas.accountQuotaName());
                System.out.println("Max value is " + quotas.max());
            }
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeAccountAttributes](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Aktualisieren von Parametern in einer DB-Parametergruppe

Das folgende Codebeispiel zeigt, wie Parameter in einer Amazon RDS-DB-Parametergruppe aktualisiert werden.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
// Modify auto_increment_offset and auto_increment_increment parameters.
public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
    try {
        Parameter parameter1 = Parameter.builder()
            .parameterName("auto_increment_offset")
            .applyMethod("immediate")
            .parameterValue("5")
            .build();

        List<Parameter> paraList = new ArrayList<>();
        paraList.add(parameter1);
        ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .parameters(paraList)
            .build();

        ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
        System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

```
}  
}
```

- Weitere API-Informationen finden Sie unter [ModifyDBParameterGroup](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Szenarien

### Erste Schritte mit DB-Instances

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 Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Führen Sie mehrere Operationen aus.

```
import com.google.gson.Gson;  
import  
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.rds.RdsClient;  
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;  
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;  
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupResponse;  
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotRequest;  
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotResponse;
```



```
import software.amazon.awssdk.services.rds.model.DBEngineVersion;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.DBParameterGroup;
import software.amazon.awssdk.services.rds.model.DBSnapshot;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsResponse;
import
    software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsResponse;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.OrderableDBInstanceOption;
import software.amazon.awssdk.services.rds.model.Parameter;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupRequest;
import
    software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbParameterGroupRequest;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
```

```

*
* https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating\_how-
services-use-secrets\_RS.html
*
* This Java example performs these tasks:
*
* 1. Returns a list of the available DB engines.
* 2. Selects an engine family and create a custom DB parameter group.
* 3. Gets the parameter groups.
* 4. Gets parameters in the group.
* 5. Modifies the auto_increment_offset parameter.
* 6. Gets and displays the updated parameters.
* 7. Gets a list of allowed engine versions.
* 8. Gets a list of micro instance classes available for the selected engine.
* 9. Creates an RDS database instance that contains a MySQL database and uses
* the parameter group.
* 10. Waits for the DB instance to be ready and prints out the connection
* endpoint value.
* 11. Creates a snapshot of the DB instance.
* 12. Waits for an RDS DB snapshot to be ready.
* 13. Deletes the RDS DB instance.
* 14. Deletes the parameter group.
*/
public class RDSScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

            Usage:
                <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier>
<dbName> <dbSnapshotIdentifier> <secretName>

            Where:
                dbGroupName - The database group name.\s
                dbParameterGroupFamily - The database parameter group name (for
example, mysql8.0).
                dbInstanceIdentifier - The database instance identifier\s
                dbName - The database name.\s
                dbSnapshotIdentifier - The snapshot identifier.\s
                secretName - The name of the AWS Secrets Manager secret that
contains the database credentials"
        """;

```

```
    if (args.length != 6) {
        System.out.println(usage);
        System.exit(1);
    }

    String dbGroupName = args[0];
    String dbParameterGroupFamily = args[1];
    String dbInstanceIdentifier = args[2];
    String dbName = args[3];
    String dbSnapshotIdentifier = args[4];
    String secretName = args[5];

    Gson gson = new Gson();
    User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
    String masterUsername = user.getUsername();
    String masterUserPassword = user.getPassword();

    Region region = Region.US_WEST_2;
    RdsClient rdsClient = RdsClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon RDS example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Return a list of the available DB engines");
    describeDBEngines(rdsClient);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Create a custom parameter group");
    createDBParameterGroup(rdsClient, dbGroupName, dbParameterGroupFamily);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. Get the parameter group");
    describeDbParameterGroups(rdsClient, dbGroupName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. Get the parameters in the group");
```

```
describeDbParameters(rdsClient, dbGroupName, 0);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Modify the auto_increment_offset parameter");
modifyDBParas(rdsClient, dbGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Display the updated value");
describeDbParameters(rdsClient, dbGroupName, -1);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Get a list of allowed engine versions");
getAllowedEngines(rdsClient, dbParameterGroupFamily);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Get a list of micro instance classes available for
the selected engine");
getMicroInstances(rdsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(
    "9. Create an RDS database instance that contains a MySQL database
and uses the parameter group");
String dbARN = createDatabaseInstance(rdsClient, dbGroupName,
dbInstanceIdentifier, dbName, masterUsername,
    masterUserPassword);
System.out.println("The ARN of the new database is " + dbARN);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Wait for DB instance to be ready");
waitForInstanceReady(rdsClient, dbInstanceIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Create a snapshot of the DB instance");
createSnapshot(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("12. Wait for DB snapshot to be ready");
waitForSnapshotReady(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Delete the DB instance");
deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Delete the parameter group");
deleteParaGroup(rdsClient, dbGroupName, dbARN);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The Scenario has successfully completed.");
System.out.println(DASHES);

rdsClient.close();
}

private static SecretsManagerClient getSecretClient() {
    Region region = Region.US_WEST_2;
    return SecretsManagerClient.builder()
        .region(region)
        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .build();
}

public static String getSecretValues(String secretName) {
    SecretsManagerClient secretClient = getSecretClient();
    GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
        .secretId(secretName)
        .build();

    GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
    return valueResponse.secretString();
}

// Delete the parameter group after database has been deleted.
// An exception is thrown if you attempt to delete the para group while database
```

```

// exists.
public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
    throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            didFind = false;
            int index = 1;
            for (DBInstance instance : instanceList) {
                instanceARN = instance.dbInstanceArn();
                if (instanceARN.compareTo(dbARN) == 0) {
                    System.out.println(dbARN + " still exists");
                    didFind = true;
                }
                if ((index == listSize) && (!didFind)) {
                    // Went through the entire list and did not find the
database ARN.
                    isDataDel = true;
                }
                Thread.sleep(sleepTime * 1000);
                index++;
            }
        }

        // Delete the para group.
        DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .build();

        rdsClient.deleteDBParameterGroup(parameterGroupRequest);
        System.out.println(dbGroupName + " was deleted.");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
    }
}

```

```
        System.exit(1);
    }
}

// Delete the DB instance.
public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .deleteAutomatedBackups(true)
            .skipFinalSnapshot(true)
            .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Waits until the snapshot instance is available.
public static void waitForSnapshotReady(RdsClient rdsClient, String
dbInstanceIdentifier,
    String dbSnapshotIdentifier) {
    try {
        boolean snapshotReady = false;
        String snapshotReadyStr;
        System.out.println("Waiting for the snapshot to become available.");

        DescribeDbSnapshotsRequest snapshotsRequest =
DescribeDbSnapshotsRequest.builder()
            .dbSnapshotIdentifier(dbSnapshotIdentifier)
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .build();

        while (!snapshotReady) {
            DescribeDbSnapshotsResponse response =
rdsClient.describeDBSnapshots(snapshotsRequest);
```

```
        List<DBSnapshot> snapshotList = response.dbSnapshots();
        for (DBSnapshot snapshot : snapshotList) {
            snapshotReadyStr = snapshot.status();
            if (snapshotReadyStr.contains("available")) {
                snapshotReady = true;
            } else {
                System.out.print(".");
                Thread.sleep(sleepTime * 1000);
            }
        }
    }

    System.out.println("The Snapshot is available!");
} catch (RdsException | InterruptedException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

// Create an Amazon RDS snapshot.
public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
    try {
        CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .dbSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
        System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
    boolean instanceReady = false;
```



```
String instanceReadyStr;
System.out.println("Waiting for instance to become available.");
try {
    DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .build();

    String endpoint = "";
    while (!instanceReady) {
        DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
        List<DBInstance> instanceList = response.dbInstances();
        for (DBInstance instance : instanceList) {
            instanceReadyStr = instance.dbInstanceStatus();
            if (instanceReadyStr.contains("available")) {
                endpoint = instance.endpoint().address();
                instanceReady = true;
            } else {
                System.out.print(".");
                Thread.sleep(sleepTime * 1000);
            }
        }
    }
    System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

} catch (RdsException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

// Create a database instance and return the ARN of the database.
public static String createDatabaseInstance(RdsClient rdsClient,
    String dbGroupName,
    String dbInstanceIdentifier,
    String dbName,
    String masterUsername,
    String masterUserPassword) {

    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
```

```
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .allocatedStorage(100)
        .dbName(dbName)
        .dbParameterGroupName(dbGroupName)
        .engine("mysql")
        .dbInstanceClass("db.m4.large")
        .engineVersion("8.0")
        .storageType("standard")
        .masterUsername(masterUsername)
        .masterUserPassword(masterUserPassword)
        .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
        return response.dbInstance().dbInstanceArn();

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }

    return "";
}

// Get a list of micro instances.
public static void getMicroInstances(RdsClient rdsClient) {
    try {
        DescribeOrderableDbInstanceOptionsRequest dbInstanceOptionsRequest =
DescribeOrderableDbInstanceOptionsRequest
            .builder()
            .engine("mysql")
            .build();

        DescribeOrderableDbInstanceOptionsResponse response = rdsClient
            .describeOrderableDBInstanceOptions(dbInstanceOptionsRequest);
        List<OrderableDBInstanceOption> orderableDBInstances =
response.orderableDBInstanceOptions();
        for (OrderableDBInstanceOption dbInstanceOption : orderableDBInstances)
        {
            System.out.println("The engine version is " +
dbInstanceOption.engineVersion());
        }
    }
}
```

```
        System.out.println("The engine description is " +
dbInstanceOption.engine());
    }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
    try {
        DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .engine("mysql")
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
        List<DBEngineVersion> dbEngines = response.dbEngineVersions();
        for (DBEngineVersion dbEngine : dbEngines) {
            System.out.println("The engine version is " +
dbEngine.engineVersion());
            System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Modify auto_increment_offset and auto_increment_increment parameters.
public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
    try {
        Parameter parameter1 = Parameter.builder()
            .parameterName("auto_increment_offset")
            .applyMethod("immediate")
            .parameterValue("5")
            .build();
```

```
        List<Parameter> paraList = new ArrayList<>();
        paraList.add(parameter1);
        ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .parameters(paraList)
            .build();

        ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
        System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Retrieve parameters in the group.
public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
int flag) {
    try {
        DescribeDbParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .build();
        } else {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .source("user")
                .build();
        }

        DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para : dbParameters) {
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            paraName = para.parameterName();
```

```
        if ((paraName.compareTo("auto_increment_offset") == 0)
            || (paraName.compareTo("auto_increment_increment ") == 0)) {
            System.out.println("*** The parameter name is " + paraName);
            System.out.println("*** The parameter value is " +
para.parameterValue());
            System.out.println("*** The parameter data type is " +
para.dataType());
            System.out.println("*** The parameter description is " +
para.description());
            System.out.println("*** The parameter allowed values is " +
para.allowedValues());
        }
    }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
    try {
        DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .maxRecords(20)
            .build();

        DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
        List<DBParameterGroup> groups = response.dbParameterGroups();
        for (DBParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbParameterGroupName());
            System.out.println("The group description is " +
group.description());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

```
public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
    try {
        CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .description("Created by using the AWS SDK for Java")
            .build();

        CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .defaultOnly(true)
            .engine("mysql")
            .maxRecords(20)
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engineOb : engines) {
            System.out.println("The name of the DB parameter group family for
the database engine is "
                + engineOb.dbParameterGroupFamily());
            System.out.println("The name of the database engine " +
engineOb.engine());
            System.out.println("The version number of the database engine " +
engineOb.engineVersion());
        }
    }
}
```

```
    }  
  
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}  
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [CreateDBInstance](#)
  - [CreateDBParameterGroup](#)
  - [CreateDBSnapshot](#)
  - [DeleteDBInstance](#)
  - [DeleteDBParameterGroup](#)
  - [DescribeDBEngineVersions](#)
  - [DescribeDBInstances](#)
  - [DescribeDBParameterGroups](#)
  - [DescribeDBParameters](#)
  - [DescribeDBSnapshots](#)
  - [DescribeOrderableDBInstanceOptions](#)
  - [ModifyDBParameterGroup](#)

## Amazon-Redshift-Beispiele mit SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon Redshift verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)

## Aktionen

Erstellen eines -Clusters

Das folgende Codebeispiel zeigt, wie Sie einen Amazon-Redshift-Cluster erstellen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Erstellen Sie den -Cluster.

```
public static void createCluster(RedshiftClient redshiftClient, String
clusterId, String masterUsername,
    String masterUserPassword) {
    try {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(masterUsername) // set the user name here
            .masterUserPassword(masterUserPassword) // set the user password
here
            .nodeType("dc2.large")
            .publiclyAccessible(true)
            .numberOfNodes(2)
            .build();

        CreateClusterResponse clusterResponse =
redshiftClient.createCluster(clusterRequest);
        System.out.println("Created cluster " +
clusterResponse.cluster().clusterIdentifier());
    }
}
```



```
    } catch (RedshiftException e) {  
  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- Weitere API-Informationen finden Sie unter [CreateCluster](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Einen Cluster löschen

Das folgende Codebeispiel zeigt, wie Sie einen Amazon-Redshift-Cluster löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Löschen Sie den Cluster.

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.redshift.RedshiftClient;  
import software.amazon.awssdk.services.redshift.model.DeleteClusterRequest;  
import software.amazon.awssdk.services.redshift.model.DeleteClusterResponse;  
import software.amazon.awssdk.services.redshift.model.RedshiftException;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class DeleteCluster {  
    public static void main(String[] args) {
```

```
final String usage = ""

    Usage:
        <clusterId>\s

    Where:
        clusterId - The id of the cluster to delete.\s
    """;

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String clusterId = args[0];
Region region = Region.US_WEST_2;
RedshiftClient redshiftClient = RedshiftClient.builder()
    .region(region)
    .build();

deleteRedshiftCluster(redshiftClient, clusterId);
redshiftClient.close();
}

public static void deleteRedshiftCluster(RedshiftClient redshiftClient, String
clusterId) {
    try {
        DeleteClusterRequest deleteClusterRequest =
DeleteClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .skipFinalClusterSnapshot(true)
            .build();

        DeleteClusterResponse response =
redshiftClient.deleteCluster(deleteClusterRequest);
        System.out.println("The status is " +
response.cluster().clusterStatus());

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [DeleteCluster](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beschreiben Ihrer Cluster

Das folgende Codebeispiel zeigt, wie Sie Ihre Amazon-Redshift-Cluster beschreiben.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Beschreiben Sie den Cluster.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.RedshiftClient;
import software.amazon.awssdk.services.redshift.model.Cluster;
import software.amazon.awssdk.services.redshift.model.DescribeClustersResponse;
import software.amazon.awssdk.services.redshift.model.RedshiftException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeClusters {

    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        RedshiftClient redshiftClient = RedshiftClient.builder()
            .region(region)
            .build();
    }
}
```

```
        describeRedshiftClusters(redshiftClient);
        redshiftClient.close();
    }

    public static void describeRedshiftClusters(RedshiftClient redshiftClient) {
        try {
            DescribeClustersResponse clusterResponse =
redshiftClient.describeClusters();
            List<Cluster> clusterList = clusterResponse.clusters();
            for (Cluster cluster : clusterList) {
                System.out.println("Cluster database name is: " + cluster.dbName());
                System.out.println("Cluster status is: " + cluster.clusterStatus());
            }

        } catch (RedshiftException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeClusters](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Ändern eines Clusters

Das folgende Codebeispiel zeigt, wie Sie einen Amazon-Redshift-Cluster ändern.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Ändern Sie einen Cluster.

```
public static void modifyCluster(RedshiftClient redshiftClient, String
clusterId) {
```

```
        try {
            ModifyClusterRequest modifyClusterRequest =
ModifyClusterRequest.builder()
                .clusterIdentifier(clusterId)
                .preferredMaintenanceWindow("wed:07:30-wed:08:00")
                .build();

            ModifyClusterResponse clusterResponse =
redshiftClient.modifyCluster(modifyClusterRequest);
            System.out.println("The modified cluster was successfully modified and
has "
                + clusterResponse.cluster().preferredMaintenanceWindow() + " as
the maintenance window");

        } catch (RedshiftException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [ModifyCluster](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Amazon Rekognition-Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon Rekognition verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Themen

- [Aktionen](#)

- [Szenarien](#)


## Aktionen

Vergleichen Sie Gesichter in einem Bild mit einem Referenzbild

Das folgende Codebeispiel zeigt, wie Gesichter in einem Bild mit einem Referenzbild mit Amazon Rekognition verglichen werden.

Weitere Informationen finden Sie unter [Vergleich von Gesichtern in Bildern](#).

SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.CompareFacesRequest;
import software.amazon.awssdk.services.rekognition.model.CompareFacesResponse;
import software.amazon.awssdk.services.rekognition.model.CompareFacesMatch;
import software.amazon.awssdk.services.rekognition.model.ComparedFace;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class CompareFaces {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <pathSource> <pathTarget>

            Where:
                pathSource - The path to the source image (for example, C:\\AWS\\
\\pic1.png).\\s
                pathTarget - The path to the target image (for example, C:\\AWS\\
\\pic2.png).\\s

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        Float similarityThreshold = 70F;
        String sourceImage = args[0];
        String targetImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        compareTwoFaces(rekClient, similarityThreshold, sourceImage, targetImage);
        rekClient.close();
    }

    public static void compareTwoFaces(RekognitionClient rekClient, Float
similarityThreshold, String sourceImage,
        String targetImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            InputStream tarStream = new FileInputStream(targetImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            SdkBytes targetBytes = SdkBytes.fromInputStream(tarStream);

            // Create an Image object for the source image.
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();
```

```

        Image tarImage = Image.builder()
            .bytes(targetBytes)
            .build();

        CompareFacesRequest facesRequest = CompareFacesRequest.builder()
            .sourceImage(souImage)
            .targetImage(tarImage)
            .similarityThreshold(similarityThreshold)
            .build();

        // Compare the two images.
        CompareFacesResponse compareFacesResult =
rekClient.compareFaces(facesRequest);
        List<CompareFacesMatch> faceDetails = compareFacesResult.faceMatches();
        for (CompareFacesMatch match : faceDetails) {
            ComparedFace face = match.face();
            BoundingBox position = face.boundingBox();
            System.out.println("Face at " + position.left().toString()
                + " " + position.top()
                + " matches with " + face.confidence().toString()
                + "% confidence.");
        }
        List<ComparedFace> uncompaired = compareFacesResult.unmatchedFaces();
        System.out.println("There was " + uncompaired.size() + " face(s) that did
not match");
        System.out.println("Source image rotation: " +
compareFacesResult.sourceImageOrientationCorrection());
        System.out.println("target image rotation: " +
compareFacesResult.targetImageOrientationCorrection());

        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println("Failed to load source image " + sourceImage);
            System.exit(1);
        }
    }
}

```

- Weitere API-Informationen finden Sie unter [CompareFaces](#) in der APIAWS SDK for Java 2.x - Referenz für .



## Eine Sammlung erstellen

Das folgende Codebeispiel zeigt, wie Sie eine Amazon Rekognition-Sammlung erstellen.

Weitere Informationen finden Sie unter [Erstellen einer Sammlung](#).

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionName>\s

                Where:
                    collectionName - The name of the collection.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String collectionId = args[0];
Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .build();

System.out.println("Creating collection: " + collectionId);
createMyCollection(rekClient, collectionId);
rekClient.close();
}

public static void createMyCollection(RekognitionClient rekClient, String
collectionId) {
    try {
        CreateCollectionRequest collectionRequest =
CreateCollectionRequest.builder()
            .collectionId(collectionId)
            .build();

        CreateCollectionResponse collectionResponse =
rekClient.createCollection(collectionRequest);
        System.out.println("CollectionArn: " +
collectionResponse.collectionArn());
        System.out.println("Status code: " +
collectionResponse.statusCode().toString());

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [CreateCollection](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen einer Sammlung.

Das folgende Codebeispiel zeigt, wie Sie eine Amazon Rekognition-Sammlung löschen.

Weitere Informationen finden Sie unter [Löschen einer Sammlung](#).

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId>\s

                Where:
                collectionId - The id of the collection to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();
```

```
        System.out.println("Deleting collection: " + collectionId);
        deleteMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    public static void deleteMyCollection(RekognitionClient rekClient, String
collectionId) {
        try {
            DeleteCollectionRequest deleteCollectionRequest =
DeleteCollectionRequest.builder()
                .collectionId(collectionId)
                .build();

            DeleteCollectionResponse deleteCollectionResponse =
rekClient.deleteCollection(deleteCollectionRequest);
            System.out.println(collectionId + ": " +
deleteCollectionResponse.statusCode().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteCollection](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen von Gesichtern aus einer Sammlung

Das folgende Codebeispiel zeigt, wie Gesichter aus einer Amazon Rekognition-Sammlung gelöscht werden.

Weitere Informationen finden Sie unter [Löschen von Gesichtern aus einer Sammlung](#).

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteFacesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteFacesFromCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId> <faceId>\s

                Where:
                    collectionId - The id of the collection from which faces are
deleted.\s

                    faceId - The id of the face to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_EAST_1;
```

```
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    System.out.println("Deleting collection: " + collectionId);
    deleteFacesCollection(rekClient, collectionId, faceId);
    rekClient.close();
}

public static void deleteFacesCollection(RekognitionClient rekClient,
    String collectionId,
    String faceId) {

    try {
        DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
            .collectionId(collectionId)
            .faceIds(faceId)
            .build();

        rekClient.deleteFaces(deleteFacesRequest);
        System.out.println("The face was deleted from the collection.");

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [DeleteFaces](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beschreiben einer Sammlung

Das folgende Codebeispiel zeigt, wie Sie eine Amazon Rekognition-Sammlung beschreiben.

Weitere Informationen finden Sie unter [Beschreiben einer Sammlung](#).

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeCollection {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <collectionName>

            Where:
                collectionName - The name of the Amazon Rekognition collection.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionName = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();
```

```
        describeColl(rekClient, collectionName);
        rekClient.close();
    }

    public static void describeColl(RekognitionClient rekClient, String
collectionName) {
        try {
            DescribeCollectionRequest describeCollectionRequest =
DescribeCollectionRequest.builder()
                .collectionId(collectionName)
                .build();

            DescribeCollectionResponse describeCollectionResponse = rekClient
                .describeCollection(describeCollectionRequest);

            System.out.println("Collection Arn : " +
describeCollectionResponse.collectionARN());
            System.out.println("Created : " +
describeCollectionResponse.creationTimestamp().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeCollection](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Gesichter in einem Bild erkennen

Das folgende Codebeispiel zeigt, wie Sie Gesichter in einem Bild mit Amazon Rekognition erkennen.

Weitere Informationen finden Sie unter [Erkennen von Gesichtern in einem Bild](#).



## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.DetectFacesRequest;
import software.amazon.awssdk.services.rekognition.model.DetectFacesResponse;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceDetail;
import software.amazon.awssdk.services.rekognition.model.AgeRange;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectFaces {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <sourceImage>

                Where:
                    sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectFacesinImage(rekClient, sourceImage);
        rekClient.close();
    }

    public static void detectFacesinImage(RekognitionClient rekClient, String
sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Create an Image object for the source image.
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            DetectFacesRequest facesRequest = DetectFacesRequest.builder()
                .attributes(Attribute.ALL)
                .image(souImage)
                .build();

            DetectFacesResponse facesResponse = rekClient.detectFaces(facesRequest);
            List<FaceDetail> faceDetails = facesResponse.faceDetails();
            for (FaceDetail face : faceDetails) {
                AgeRange ageRange = face.ageRange();
                System.out.println("The detected face is estimated to be between "
                    + ageRange.low().toString() + " and " +
ageRange.high().toString()
                    + " years old.");

                System.out.println("There is a smile : " +
face.smile().value().toString());
            }
        }
    }
}
```

```
        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [DetectFaces](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erkennen von Labels in einem Bild

Das folgende Codebeispiel zeigt, wie Sie Labels in einem Bild mit Amazon Rekognition erkennen.

Weitere Informationen finden Sie unter [Erkennen von Labels in einem Bild](#).

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsRequest;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DetectLabels {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectImageLabels(rekClient, sourceImage);
        rekClient.close();
    }

    public static void detectImageLabels(RekognitionClient rekClient, String
sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Create an Image object for the source image.
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            DetectLabelsRequest detectLabelsRequest = DetectLabelsRequest.builder()
                .image(souImage)
                .maxLabels(10)
```

```
        .build();

        DetectLabelsResponse labelsResponse =
rekClient.detectLabels(detectLabelsRequest);
        List<Label> labels = labelsResponse.labels();
        System.out.println("Detected labels for the given photo");
        for (Label label : labels) {
            System.out.println(label.name() + ": " +
label.confidence().toString());
        }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [DetectLabels](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erkennen von Labels zur Moderation in einem Bild

Das folgende Codebeispiel zeigt, wie Sie Moderationslabels in einem Bild mit Amazon Rekognition erkennen. Moderationslabels kennzeichnen Inhalte, die für einige Zielgruppen möglicherweise unangemessen sind.

Weitere Informationen finden Sie unter [Erkennen von unangemessenen Bildern](#).

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
```

```
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import
    software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.ModerationLabel;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectModerationLabels {

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
                """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectModLabels(rekClient, sourceImage);
    }
}
```

```

        rekClient.close();
    }

    public static void detectModLabels(RekognitionClient rekClient, String
sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            DetectModerationLabelsRequest moderationLabelsRequest =
DetectModerationLabelsRequest.builder()
                .image(souImage)
                .minConfidence(60F)
                .build();

            DetectModerationLabelsResponse moderationLabelsResponse = rekClient
                .detectModerationLabels(moderationLabelsRequest);
            List<ModerationLabel> labels =
moderationLabelsResponse.moderationLabels();
            System.out.println("Detected labels for image");
            for (ModerationLabel label : labels) {
                System.out.println("Label: " + label.name()
                    + "\n Confidence: " + label.confidence().toString() + "%"
                    + "\n Parent:" + label.parentName());
            }

        } catch (RekognitionException | FileNotFoundException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}

```

- Weitere API-Informationen finden Sie unter [DetectModerationLabels](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erkennen von Text in einem Bild

Das folgende Codebeispiel zeigt, wie Sie Text in einem Bild mit Amazon Rekognition erkennen.

Weitere Informationen finden Sie unter [Erkennen von Text in einem Bild](#).

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DetectTextRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectTextResponse;
import software.amazon.awssdk.services.rekognition.model.TextDetection;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectText {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <sourceImage>

                Where:
                    sourceImage - The path to the image that contains text (for
example, C:\\AWS\\pic1.png).\\s
                """;
```



```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectTextLabels(rekClient, sourceImage);
        rekClient.close();
    }

    public static void detectTextLabels(RekognitionClient rekClient, String
sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            DetectTextRequest textRequest = DetectTextRequest.builder()
                .image(souImage)
                .build();

            DetectTextResponse textResponse = rekClient.detectText(textRequest);
            List<TextDetection> textCollection = textResponse.textDetections();
            System.out.println("Detected lines and words");
            for (TextDetection text : textCollection) {
                System.out.println("Detected: " + text.detectedText());
                System.out.println("Confidence: " + text.confidence().toString());
                System.out.println("Id : " + text.id());
                System.out.println("Parent Id: " + text.parentId());
                System.out.println("Type: " + text.type());
                System.out.println();
            }
        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

```
}  
}
```

- Weitere API-Informationen finden Sie unter [DetectText](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Gesichter einer Sammlung zuordnen

Das folgende Codebeispiel zeigt, wie Gesichter in einem Bild indiziert und zu einer Amazon Rekognition-Sammlung hinzugefügt werden.

Weitere Informationen finden Sie unter [Hinzufügen von Gesichtern zu einer Sammlung](#).

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.rekognition.RekognitionClient;  
import software.amazon.awssdk.services.rekognition.model.IndexFacesResponse;  
import software.amazon.awssdk.services.rekognition.model.IndexFacesRequest;  
import software.amazon.awssdk.services.rekognition.model.Image;  
import software.amazon.awssdk.services.rekognition.model.QualityFilter;  
import software.amazon.awssdk.services.rekognition.model.Attribute;  
import software.amazon.awssdk.services.rekognition.model.FaceRecord;  
import software.amazon.awssdk.services.rekognition.model.UnindexedFace;  
import software.amazon.awssdk.services.rekognition.model.RekognitionException;  
import software.amazon.awssdk.services.rekognition.model.Reason;  
import java.io.FileInputStream;  
import java.io.FileNotFoundException;  
import java.io.InputStream;  
import java.util.List;  
  
/**  
 * Before running this Java V2 code example, set up your development
```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class AddFacesToCollection {
    public static void main(String[] args) {

        final String usage = ""

                Usage:      <collectionId> <sourceImage>

                Where:
                    collectionName - The name of the collection.
                    sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        addToCollection(rekClient, collectionId, sourceImage);
        rekClient.close();
    }

    public static void addToCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                    .bytes(sourceBytes)
                    .build();
        }
    }
}
```

```

        IndexFacesRequest facesRequest = IndexFacesRequest.builder()
            .collectionId(collectionId)
            .image(souImage)
            .maxFaces(1)
            .qualityFilter(QualityFilter.AUTO)
            .detectionAttributes(Attribute.DEFAULT)
            .build();

        IndexFacesResponse facesResponse = rekClient.indexFaces(facesRequest);
        System.out.println("Results for the image");
        System.out.println("\n Faces indexed:");
        List<FaceRecord> faceRecords = facesResponse.faceRecords();
        for (FaceRecord faceRecord : faceRecords) {
            System.out.println("  Face ID: " + faceRecord.face().faceId());
            System.out.println("  Location:" +
faceRecord.faceDetail().boundingBox().toString());
        }

        List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
        System.out.println("Faces not indexed:");
        for (UnindexedFace unindexedFace : unindexedFaces) {
            System.out.println("  Location:" +
unindexedFace.faceDetail().boundingBox().toString());
            System.out.println("  Reasons:");
            for (Reason reason : unindexedFace.reasons()) {
                System.out.println("Reason: " + reason);
            }
        }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}

```

- Weitere API-Informationen finden Sie unter [IndexFaces](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Sammlungen

Das folgende Codebeispiel zeigt, wie Sie Amazon Rekognition-Sammlungen auflisten.

Weitere Informationen finden Sie unter [Sammlungen auflisten](#).

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsRequest;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCollections {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        System.out.println("Listing collections");
        listAllCollections(rekClient);
        rekClient.close();
    }

    public static void listAllCollections(RekognitionClient rekClient) {
        try {
            ListCollectionsRequest listCollectionsRequest =
ListCollectionsRequest.builder()
                .maxResults(10)
```

```
        .build();

        ListCollectionsResponse response =
rekClient.listCollections(listCollectionsRequest);
        List<String> collectionIds = response.collectionIds();
        for (String resultId : collectionIds) {
            System.out.println(resultId);
        }

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [ListCollections](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Gesichtern in einer Sammlung

Das folgende Codebeispiel zeigt, wie Gesichter in einer Amazon Rekognition-Sammlung aufgelistet werden.

Weitere Informationen finden Sie unter [Gesichter in einer Sammlung auflisten](#).

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Face;
import software.amazon.awssdk.services.rekognition.model.ListFacesRequest;
import software.amazon.awssdk.services.rekognition.model.ListFacesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
```

```
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListFacesInCollection {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <collectionId>

            Where:
                collectionId - The name of the collection.\s
            """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        System.out.println("Faces in collection " + collectionId);
        listFacesCollection(rekClient, collectionId);
        rekClient.close();
    }

    public static void listFacesCollection(RekognitionClient rekClient, String
collectionId) {
        try {
            ListFacesRequest facesRequest = ListFacesRequest.builder()
                .collectionId(collectionId)
                .maxResults(10)
                .build();
        }
    }
}
```

```
        ListFacesResponse facesResponse = rekClient.listFaces(facesRequest);
        List<Face> faces = facesResponse.faces();
        for (Face face : faces) {
            System.out.println("Confidence level there is a face: " +
face.confidence());
            System.out.println("The face Id value is " + face.faceId());
        }

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [ListFaces](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erkennen von Prominenten in einem Bild

Das folgende Codebeispiel zeigt, wie Prominente in einem Bild mit Amazon Rekognition erkannt werden.

Weitere Informationen finden Sie unter [Erkennen von Prominenten in einem Bild](#).

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
```



```
import
    software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesRequest;
import
    software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Celebrity;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RecognizeCelebrities {
    public static void main(String[] args) {
        final String usage = ""
            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        System.out.println("Locating celebrities in " + sourceImage);
        recognizeAllCelebrities(rekClient, sourceImage);
        rekClient.close();
    }

    public static void recognizeAllCelebrities(RekognitionClient rekClient, String
sourceImage) {
```

```
try {
    InputStream sourceStream = new FileInputStream(sourceImage);
    SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
    Image souImage = Image.builder()
        .bytes(sourceBytes)
        .build();

    RecognizeCelebritiesRequest request =
RecognizeCelebritiesRequest.builder()
        .image(souImage)
        .build();

    RecognizeCelebritiesResponse result =
rekClient.recognizeCelebrities(request);
    List<Celebrity> celebs = result.celebrityFaces();
    System.out.println(celebs.size() + " celebrity(s) were recognized.\n");
    for (Celebrity celebrity : celebs) {
        System.out.println("Celebrity recognized: " + celebrity.name());
        System.out.println("Celebrity ID: " + celebrity.id());

        System.out.println("Further information (if available):");
        for (String url : celebrity.urls()) {
            System.out.println(url);
        }
        System.out.println();
    }
    System.out.println(result.unrecognizedFaces().size() + " face(s) were
unrecognized.");

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [RecognizeCelebrities](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Suche nach Gesichtern in einer Sammlung

Das folgende Codebeispiel zeigt, wie Sie in einer Amazon Rekognition-Sammlung nach Gesichtern suchen, die mit einem anderen Gesicht aus der Sammlung übereinstimmen.

Weitere Informationen finden Sie unter [Nach einem Gesicht suchen \(Gesichts-ID\)](#).

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SearchFaceMatchingImageCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId> <sourceImage>
```

```
        Where:
            collectionId - The id of the collection. \s
            sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png)\\.s

        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String collectionId = args[0];
    String sourceImage = args[1];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    System.out.println("Searching for a face in a collections");
    searchFaceInCollection(rekClient, collectionId, sourceImage);
    rekClient.close();
}

public static void searchFaceInCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {
    try {
        InputStream sourceStream = new FileInputStream(new File(sourceImage));
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        SearchFacesByImageRequest facesByImageRequest =
SearchFacesByImageRequest.builder()
            .image(souImage)
            .maxFaces(10)
            .faceMatchThreshold(70F)
            .collectionId(collectionId)
            .build();

        SearchFacesByImageResponse imageResponse =
rekClient.searchFacesByImage(facesByImageRequest);
        System.out.println("Faces matching in the collection");
    }
}
```

```
        List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
        for (FaceMatch face : faceImageMatches) {
            System.out.println("The similarity level is " + face.similarity());
            System.out.println();
        }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [SearchFaces](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Suche nach Gesichtern in einer Sammlung im Vergleich zu einem Referenzbild

Das folgende Codebeispiel zeigt, wie Sie in einer Amazon Rekognition-Sammlung im Vergleich zu einem Referenzbild nach Gesichtern suchen.

Weitere Informationen finden Sie unter [Nach einem Gesicht suchen \(Bild\)](#).

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.SearchFacesRequest;
import software.amazon.awssdk.services.rekognition.model.SearchFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class SearchFaceMatchingIdCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId> <sourceImage>

                Where:
                    collectionId - The id of the collection. \s
                    sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        System.out.println("Searching for a face in a collections");
        searchFaceById(rekClient, collectionId, faceId);
        rekClient.close();
    }

    public static void searchFaceById(RekognitionClient rekClient, String
collectionId, String faceId) {
        try {
            SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
                .collectionId(collectionId)
                .faceId(faceId)
                .faceMatchThreshold(70F)
                .maxFaces(2)
```

```
        .build();

        SearchFacesResponse imageResponse =
rekClient.searchFaces(searchFacesRequest);
        System.out.println("Faces matching in the collection");
        List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
        for (FaceMatch face : faceImageMatches) {
            System.out.println("The similarity level is " + face.similarity());
            System.out.println();
        }

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [SearchFacesByImage](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Szenarien

### 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 Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Abrufen von Informationen aus einem Video, das sich in einem Amazon-S3-Bucket befindet.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognitionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognition;
import software.amazon.awssdk.services.rekognition.model.CelebrityDetail;
import
    software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionRequest;
import
    software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionRequest;
import
    software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionResponse;
import java.util.List;

/**
 * To run this code example, ensure that you perform the Prerequisites as stated
 * in the Amazon Rekognition Guide:
 * https://docs.aws.amazon.com/rekognition/latest/dg/video-analyzing-with-sqs.html
 *
 * Also, ensure that set up your development environment, including your
 * credentials.
 *
 * For information, see this documentation topic:
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class VideoCelebrityDetection {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
```



```

        bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
        video - The name of video (for example, people.mp4).\s
        topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
        roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
        """;

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucket = args[0];
    String video = args[1];
    String topicArn = args[2];
    String roleArn = args[3];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    NotificationChannel channel = NotificationChannel.builder()
        .snsTopicArn(topicArn)
        .roleArn(roleArn)
        .build();

    startCelebrityDetection(rekClient, channel, bucket, video);
    getCelebrityDetectionResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

public static void startCelebrityDetection(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();
    }
}

```

```
        Video vid0b = Video.builder()
            .s3object(s3obj)
            .build();

        StartCelebrityRecognitionRequest recognitionRequest =
StartCelebrityRecognitionRequest.builder()
            .jobTag("Celebrities")
            .notificationChannel(channel)
            .video(vid0b)
            .build();

        StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
rekClient
            .startCelebrityRecognition(recognitionRequest);
        startJobId = startCelebrityRecognitionResult.jobId();

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void getCelebrityDetectionResults(RekognitionClient rekClient) {

    try {
        String paginationToken = null;
        GetCelebrityRecognitionResponse recognitionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (recognitionResponse != null)
                paginationToken = recognitionResponse.nextToken();

            GetCelebrityRecognitionRequest recognitionRequest =
GetCelebrityRecognitionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
                .maxResults(10)
                .build();

            // Wait until the job succeeds
```

```
        while (!finished) {
            recognitionResponse =
rekClient.getCelebrityRecognition(recognitionRequest);
            status = recognitionResponse.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(yy + " status is: " + status);
                Thread.sleep(1000);
            }
            yy++;
        }

        finished = false;

        // Proceed when the job is done - otherwise VideoMetadata is null.
        VideoMetadata videoMetaData = recognitionResponse.videoMetadata();
        System.out.println("Format: " + videoMetaData.format());
        System.out.println("Codec: " + videoMetaData.codec());
        System.out.println("Duration: " + videoMetaData.durationMillis());
        System.out.println("FrameRate: " + videoMetaData.frameRate());
        System.out.println("Job");

        List<CelebrityRecognition> celebs =
rekognitionResponse.celebrities();
        for (CelebrityRecognition celeb : celebs) {
            long seconds = celeb.timestamp() / 1000;
            System.out.print("Sec: " + seconds + " ");
            CelebrityDetail details = celeb.celebrity();
            System.out.println("Name: " + details.name());
            System.out.println("Id: " + details.id());
            System.out.println();
        }

        } while (recognitionResponse.nextToken() != null);

    } catch (RekognitionException | InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

Erkennen Sie Labels in einem Video mithilfe einer Labelerkennung.

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
```

```
final String usage = ""

    Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

    Where:
        bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
        video - The name of the video (for example, people.mp4).\s
        queueUrl- The URL of a SQS queue.\s
        topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
        roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
    """;

if (args.length != 5) {
    System.out.println(usage);
    System.exit(1);
}

String bucket = args[0];
String video = args[1];
String queueUrl = args[2];
String topicArn = args[3];
String roleArn = args[4];
Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .build();

SqsClient sqs = SqsClient.builder()
    .region(Region.US_EAST_1)
    .build();

NotificationChannel channel = NotificationChannel.builder()
    .snsTopicArn(topicArn)
    .roleArn(roleArn)
    .build();

startLabels(rekClient, channel, bucket, video);
getLabelJob(rekClient, sqs, queueUrl);
System.out.println("This example is done!");
sqs.close();
rekClient.close();
```

```
}

public static void startLabels(RecognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
            .s3Object(s3obj)
            .build();

        StartLabelDetectionRequest labelDetectionRequest =
        StartLabelDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vidObj)
            .minConfidence(50F)
            .build();

        StartLabelDetectionResponse labelDetectionResponse =
        rekClient.startLabelDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();

        boolean ans = true;
        String status = "";
        int yy = 0;
        while (ans) {

            GetLabelDetectionRequest detectionRequest =
            GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .maxResults(10)
                .build();

            GetLabelDetectionResponse result =
            rekClient.getLabelDetection(detectionRequest);
            status = result.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
```

```
        ans = false;
    else
        System.out.println(yy + " status is: " + status);

        Thread.sleep(1000);
        yy++;
    }

    System.out.println(startJobId + " status is: " + status);

} catch (RekognitionException | InterruptedException e) {
    e.getMessage();
    System.exit(1);
}
}

public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
    List<Message> messages;
    ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();

    try {
        messages = sqs.receiveMessage(messageRequest).messages();

        if (!messages.isEmpty()) {
            for (Message message : messages) {
                String notification = message.body();

                // Get the status and job id from the notification
                ObjectMapper mapper = new ObjectMapper();
                JsonNode jsonMessageTree = mapper.readTree(notification);
                JsonNode messageBodyText = jsonMessageTree.get("Message");
                ObjectMapper operationResultMapper = new ObjectMapper();
                JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                JsonNode operationJobId = jsonResultTree.get("JobId");
                JsonNode operationStatus = jsonResultTree.get("Status");
                System.out.println("Job found in JSON is " + operationJobId);

                DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                    .queueUrl(queueUrl)
```

```
        .build();

        String jobId = operationJobId.textValue();
        if (startJobId.compareTo(jobId) == 0) {
            System.out.println("Job id: " + operationJobId);
            System.out.println("Status : " +
operationStatus.toString());

            if (operationStatus.asText().equals("SUCCEEDED"))
                getResultsLabels(rekClient);
            else
                System.out.println("Video analysis failed");

            sqs.deleteMessage(deleteMessageRequest);
        } else {
            System.out.println("Job received was not job " +
startJobId);

            sqs.deleteMessage(deleteMessageRequest);
        }
    }
}

} catch (RekognitionException e) {
    e.getMessage();
    System.exit(1);
} catch (JsonMappingException e) {
    e.printStackTrace();
} catch (JsonProcessingException e) {
    e.printStackTrace();
}
}

// Gets the job results by calling GetLabelDetection
private static void getResultsLabels(RekognitionClient rekClient) {

    int maxResults = 10;
    String paginationToken = null;
    GetLabelDetectionResponse labelDetectionResult = null;

    try {
        do {
            if (labelDetectionResult != null)
                paginationToken = labelDetectionResult.nextToken();
        }
    }
}
```



```
GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
    .jobId(startJobId)
    .sortBy(LabelDetectionSortBy.TIMESTAMP)
    .maxResults(maxResults)
    .nextToken(paginationToken)
    .build();

LabelDetectionResult labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
VideoMetadata videoMetaData = labelDetectionResult.videoMetadata();
System.out.println("Format: " + videoMetaData.format());
System.out.println("Codec: " + videoMetaData.codec());
System.out.println("Duration: " + videoMetaData.durationMillis());
System.out.println("FrameRate: " + videoMetaData.frameRate());

List<LabelDetection> detectedLabels = labelDetectionResult.labels();
for (LabelDetection detectedLabel : detectedLabels) {
    long seconds = detectedLabel.timestamp();
    Label label = detectedLabel.label();
    System.out.println("Millisecond: " + seconds + " ");

    System.out.println("    Label:" + label.name());
    System.out.println("    Confidence:" +
detectedLabel.label().confidence().toString());

    List<Instance> instances = label.instances();
    System.out.println("    Instances of " + label.name());

    if (instances.isEmpty()) {
        System.out.println("        " + "None");
    } else {
        for (Instance instance : instances) {
            System.out.println("            Confidence: " +
instance.confidence().toString());
            System.out.println("            Bounding box: " +
instance.boundingBox().toString());
        }
    }
    System.out.println("    Parent labels for " + label.name() +
":");

    List<Parent> parents = label.parents();

    if (parents.isEmpty()) {
```

```

        System.out.println("        None");
    } else {
        for (Parent parent : parents) {
            System.out.println("        " + parent.name());
        }
    }
    System.out.println();
}
} while (labelDetectionResult != null &&
labelDetectionResult.nextToken() != null);

} catch (RekognitionException e) {
    e.getMessage();
    System.exit(1);
}
}
}

```

Erkennen von Gesichtern in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;

```

```
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of the video (for example, people.mp4).\s
                queueUrl- The URL of a SQS queue.\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String queueUrl = args[2];
        String topicArn = args[3];
        String roleArn = args[4];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
```

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

    SqsClient sqs = SqsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    NotificationChannel channel = NotificationChannel.builder()
        .snsTopicArn(topicArn)
        .roleArn(roleArn)
        .build();

    startLabels(rekClient, channel, bucket, video);
    getLabelJob(rekClient, sqs, queueUrl);
    System.out.println("This example is done!");
    sqs.close();
    rekClient.close();
}

public static void startLabels(RecognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
            .s3Object(s3obj)
            .build();

        StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vidObj)
            .minConfidence(50F)
            .build();

        StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();
    }
}
```

```
        boolean ans = true;
        String status = "";
        int yy = 0;
        while (ans) {

            GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .maxResults(10)
                .build();

            GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
            status = result.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                ans = false;
            else
                System.out.println(yy + " status is: " + status);

            Thread.sleep(1000);
            yy++;
        }

        System.out.println(startJobId + " status is: " + status);

    } catch (RekognitionException | InterruptedException e) {
        e.getMessage();
        System.exit(1);
    }
}

public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
    List<Message> messages;
    ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();

    try {
        messages = sqs.receiveMessage(messageRequest).messages();

        if (!messages.isEmpty()) {
```

```
        for (Message message : messages) {
            String notification = message.body();

            // Get the status and job id from the notification
            ObjectMapper mapper = new ObjectMapper();
            JsonNode jsonMessageTree = mapper.readTree(notification);
            JsonNode messageBodyText = jsonMessageTree.get("Message");
            ObjectMapper operationResultMapper = new ObjectMapper();
            JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
            JsonNode operationJobId = jsonResultTree.get("JobId");
            JsonNode operationStatus = jsonResultTree.get("Status");
            System.out.println("Job found in JSON is " + operationJobId);

            DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                .queueUrl(queueUrl)
                .build();

            String jobId = operationJobId.textValue();
            if (startJobId.compareTo(jobId) == 0) {
                System.out.println("Job id: " + operationJobId);
                System.out.println("Status : " +
operationStatus.toString());

                if (operationStatus.asText().equals("SUCCEEDED"))
                    getResultsLabels(rekClient);
                else
                    System.out.println("Video analysis failed");

                sqs.deleteMessage(deleteMessageRequest);
            } else {
                System.out.println("Job received was not job " +
startJobId);
                sqs.deleteMessage(deleteMessageRequest);
            }
        }
    }

} catch (RekognitionException e) {
    e.getMessage();
    System.exit(1);
} catch (JsonMappingException e) {
    e.printStackTrace();
}
```

```
    } catch (JsonProcessingException e) {
        e.printStackTrace();
    }
}

// Gets the job results by calling GetLabelDetection
private static void getResultsLabels(RekognitionClient rekClient) {

    int maxResults = 10;
    String paginationToken = null;
    GetLabelDetectionResponse labelDetectionResult = null;

    try {
        do {
            if (labelDetectionResult != null)
                paginationToken = labelDetectionResult.nextToken();

            GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .sortBy(LabelDetectionSortBy.TIMESTAMP)
                .maxResults(maxResults)
                .nextToken(paginationToken)
                .build();

            labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
            VideoMetadata videoMetaData = labelDetectionResult.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());

            List<LabelDetection> detectedLabels = labelDetectionResult.labels();
            for (LabelDetection detectedLabel : detectedLabels) {
                long seconds = detectedLabel.timestamp();
                Label label = detectedLabel.label();
                System.out.println("Millisecond: " + seconds + " ");

                System.out.println("  Label:" + label.name());
                System.out.println("  Confidence:" +
detectedLabel.label().confidence().toString());

                List<Instance> instances = label.instances();
```

```

        System.out.println("  Instances of " + label.name());

        if (instances.isEmpty()) {
            System.out.println("    " + "None");
        } else {
            for (Instance instance : instances) {
                System.out.println("      Confidence: " +
instance.confidence().toString());
                System.out.println("      Bounding box: " +
instance.boundingBox().toString());
            }
        }
        System.out.println("  Parent labels for " + label.name() +
":");

        List<Parent> parents = label.parents();

        if (parents.isEmpty()) {
            System.out.println("    None");
        } else {
            for (Parent parent : parents) {
                System.out.println("      " + parent.name());
            }
        }
        System.out.println();
    }
    } while (labelDetectionResult != null &&
labelDetectionResult.nextToken() != null);

    } catch (RekognitionException e) {
        e.getMessage();
        System.exit(1);
    }
}
}

```

Erkennen von unangemessenen oder anstößigen Inhalten in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;

```



```
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartContentModerationRequest;
import
    software.amazon.awssdk.services.rekognition.model.StartContentModerationResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.GetContentModerationResponse;
import
    software.amazon.awssdk.services.rekognition.model.GetContentModerationRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.ContentModerationDetection;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetectInappropriate {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String bucket = args[0];
    String video = args[1];
    String topicArn = args[2];
    String roleArn = args[3];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    NotificationChannel channel = NotificationChannel.builder()
        .snsTopicArn(topicArn)
        .roleArn(roleArn)
        .build();

    startModerationDetection(rekClient, channel, bucket, video);
    getModResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

public static void startModerationDetection(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {

    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
            .jobTag("Moderation")
            .notificationChannel(channel)
            .video(vidObj)
            .build();
```

```
        StartContentModerationResponse startModDetectionResult = rekClient
            .startContentModeration(modDetectionRequest);
        startJobId = startModDetectionResult.jobId();

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void getModResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetContentModerationResponse modDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (modDetectionResponse != null)
                paginationToken = modDetectionResponse.nextToken();

            GetContentModerationRequest modRequest =
GetContentModerationRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds.
            while (!finished) {
                modDetectionResponse =
rekClient.getContentModeration(modRequest);
                status = modDetectionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }
        }
    }
}
```

```

        finished = false;

        // Proceed when the job is done - otherwise VideoMetadata is null.
        VideoMetadata videoMetadata = modDetectionResponse.videoMetadata();
        System.out.println("Format: " + videoMetadata.format());
        System.out.println("Codec: " + videoMetadata.codec());
        System.out.println("Duration: " + videoMetadata.durationMillis());
        System.out.println("FrameRate: " + videoMetadata.frameRate());
        System.out.println("Job");

        List<ContentModerationDetection> mods =
modDetectionResponse.moderationLabels();
        for (ContentModerationDetection mod : mods) {
            long seconds = mod.timestamp() / 1000;
            System.out.print("Mod label: " + seconds + " ");
            System.out.println(mod.moderationLabel().toString());
            System.out.println();
        }

        } while (modDetectionResponse != null &&
modDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}

```

Erkennen Sie technische Signal-Segmente und Einstellungserkennungssegmente in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartShotDetectionFilter;
import
    software.amazon.awssdk.services.rekognition.model.StartTechnicalCueDetectionFilter;
import
    software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionFilters;

```

```
import
    software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionRequest;
import
    software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.SegmentDetection;
import software.amazon.awssdk.services.rekognition.model.TechnicalCueSegment;
import software.amazon.awssdk.services.rekognition.model.ShotSegment;
import software.amazon.awssdk.services.rekognition.model.SegmentType;
import software.amazon.awssdk.services.sqs.SqsClient;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetectSegment {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String bucket = args[0];
    String video = args[1];
    String topicArn = args[2];
    String roleArn = args[3];

    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    SqsClient sqs = SqsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    NotificationChannel channel = NotificationChannel.builder()
        .snsTopicArn(topicArn)
        .roleArn(roleArn)
        .build();

    startSegmentDetection(rekClient, channel, bucket, video);
    getSegmentResults(rekClient);
    System.out.println("This example is done!");
    sqs.close();
    rekClient.close();
}

public static void startSegmentDetection(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
            .s3Object(s3obj)
            .build();
```

```
        StartShotDetectionFilter cueDetectionFilter =
StartShotDetectionFilter.builder()
        .minSegmentConfidence(60F)
        .build();

        StartTechnicalCueDetectionFilter technicalCueDetectionFilter =
StartTechnicalCueDetectionFilter.builder()
        .minSegmentConfidence(60F)
        .build();

        StartSegmentDetectionFilters filters =
StartSegmentDetectionFilters.builder()
        .shotFilter(cueDetectionFilter)
        .technicalCueFilter(technicalCueDetectionFilter)
        .build();

        StartSegmentDetectionRequest segDetectionRequest =
StartSegmentDetectionRequest.builder()
        .jobTag("DetectingLabels")
        .notificationChannel(channel)
        .segmentTypes(SegmentType.TECHNICAL_CUE, SegmentType.SHOT)
        .video(vid0b)
        .filters(filters)
        .build();

        StartSegmentDetectionResponse segDetectionResponse =
rekClient.startSegmentDetection(segDetectionRequest);
        startJobId = segDetectionResponse.jobId();

    } catch (RekognitionException e) {
        e.getMessage();
        System.exit(1);
    }
}

public static void getSegmentResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetSegmentDetectionResponse segDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
```

```
        if (segDetectionResponse != null)
            paginationToken = segDetectionResponse.nextToken();

        GetSegmentDetectionRequest recognitionRequest =
GetSegmentDetectionRequest.builder()
            .jobId(startJobId)
            .nextToken(paginationToken)
            .maxResults(10)
            .build();

        // Wait until the job succeeds.
        while (!finished) {
            segDetectionResponse =
rekClient.getSegmentDetection(recognitionRequest);
            status = segDetectionResponse.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(yy + " status is: " + status);
                Thread.sleep(1000);
            }
            yy++;
        }
        finished = false;

        // Proceed when the job is done - otherwise VideoMetadata is null.
        List<VideoMetadata> videoMetaData =
segDetectionResponse.videoMetadata();
        for (VideoMetadata metaData : videoMetaData) {
            System.out.println("Format: " + metaData.format());
            System.out.println("Codec: " + metaData.codec());
            System.out.println("Duration: " + metaData.durationMillis());
            System.out.println("FrameRate: " + metaData.frameRate());
            System.out.println("Job");
        }

        List<SegmentDetection> detectedSegments =
segDetectionResponse.segments();
        for (SegmentDetection detectedSegment : detectedSegments) {
            String type = detectedSegment.type().toString();
            if (type.contains(SegmentType.TECHNICAL_CUE.toString())) {
                System.out.println("Technical Cue");
            }
        }
    }
}
```



```

        TechnicalCueSegment segmentCue =
detectedSegment.technicalCueSegment();
        System.out.println("\tType: " + segmentCue.type());
        System.out.println("\tConfidence: " +
segmentCue.confidence().toString());
    }

    if (type.contains(SegmentType.SHOT.toString())) {
        System.out.println("Shot");
        ShotSegment segmentShot = detectedSegment.shotSegment();
        System.out.println("\tIndex " + segmentShot.index());
        System.out.println("\tConfidence: " +
segmentShot.confidence().toString());
    }

    long seconds = detectedSegment.durationMillis();
    System.out.println("\tDuration : " + seconds + " milliseconds");
    System.out.println("\tStart time code: " +
detectedSegment.startTimecodeSMPTE());
    System.out.println("\tEnd time code: " +
detectedSegment.endTimecodeSMPTE());
    System.out.println("\tDuration time code: " +
detectedSegment.durationSMPTE());
    System.out.println();
}

} while (segDetectionResponse != null &&
segDetectionResponse.nextToken() != null);

} catch (RekognitionException | InterruptedException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
}

```

Erkennen Sie Text in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;

```

```
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.StartTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.TextDetectionResult;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetectText {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
```

```
String roleArn = args[3];

Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .build();

NotificationChannel channel = NotificationChannel.builder()
    .snsTopicArn(topicArn)
    .roleArn(roleArn)
    .build();

startTextLabels(rekClient, channel, bucket, video);
getTextResults(rekClient);
System.out.println("This example is done!");
rekClient.close();
}

public static void startTextLabels(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vidObj)
            .build();

        StartTextDetectionResponse labelDetectionResponse =
rekClient.startTextDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();

    } catch (RekognitionException e) {
```

```
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void getTextResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetTextDetectionResponse textDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (textDetectionResponse != null)
                paginationToken = textDetectionResponse.nextToken();

            GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds.
            while (!finished) {
                textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
                status = textDetectionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }

            finished = false;

            // Proceed when the job is done - otherwise VideoMetadata is null.
            VideoMetadata videoMetaData = textDetectionResponse.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
        }
    }
}
```

```

        System.out.println("Codec: " + videoMetaData.codec());
        System.out.println("Duration: " + videoMetaData.durationMillis());
        System.out.println("FrameRate: " + videoMetaData.frameRate());
        System.out.println("Job");

        List<TextDetectionResult> labels =
textDetectionResponse.textDetections();
        for (TextDetectionResult detectedText : labels) {
            System.out.println("Confidence: " +
detectedText.textDetection().confidence().toString());
            System.out.println("Id : " + detectedText.textDetection().id());
            System.out.println("Parent Id: " +
detectedText.textDetection().parentId());
            System.out.println("Type: " +
detectedText.textDetection().type());
            System.out.println("Text: " +
detectedText.textDetection().detectedText());
            System.out.println();
        }

        } while (textDetectionResponse != null &&
textDetectionResponse.nextToken() != null);

    } catch (RekognitionException | InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

Erkennen Sie Personen in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.StartPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.GetPersonTrackingResponse;

```

```
import software.amazon.awssdk.services.rekognition.model.GetPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.PersonDetection;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoPersonDetection {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
```

```
        .build();

        NotificationChannel channel = NotificationChannel.builder()
            .snsTopicArn(topicArn)
            .roleArn(roleArn)
            .build();

        startPersonLabels(rekClient, channel, bucket, video);
        getPersonDetectionResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startPersonLabels(RecognitionClient rekClient,
        NotificationChannel channel,
        String bucket,
        String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                .bucket(bucket)
                .name(video)
                .build();

            Video vidObj = Video.builder()
                .s3Object(s3Obj)
                .build();

            StartPersonTrackingRequest personTrackingRequest =
                StartPersonTrackingRequest.builder()
                    .jobTag("DetectingLabels")
                    .video(vidObj)
                    .notificationChannel(channel)
                    .build();

            StartPersonTrackingResponse labelDetectionResponse =
                rekClient.startPersonTracking(personTrackingRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RecognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

```
public static void getPersonDetectionResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetPersonTrackingResponse personTrackingResult = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (personTrackingResult != null)
                paginationToken = personTrackingResult.nextToken();

            GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds
            while (!finished) {

                personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
                status = personTrackingResult.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }

            finished = false;

            // Proceed when the job is done - otherwise VideoMetadata is null.
            VideoMetadata videoMetaData = personTrackingResult.videoMetadata();

            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());
        }
    }
}
```



```
        System.out.println("Job");

        List<PersonDetection> detectedPersons =
personTrackingResult.persons();
        for (PersonDetection detectedPerson : detectedPersons) {
            long seconds = detectedPerson.timestamp() / 1000;
            System.out.print("Sec: " + seconds + " ");
            System.out.println("Person Identifier: " +
detectedPerson.person().index());
            System.out.println();
        }

        } while (personTrackingResult != null &&
personTrackingResult.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [GetCelebrityRecognition](#)
  - [GetContentModeration](#)
  - [GetLabelDetection](#)
  - [GetPersonTracking](#)
  - [GetSegmentDetection](#)
  - [GetTextDetection](#)
  - [StartCelebrityRecognition](#)
  - [StartContentModeration](#)
  - [StartLabelDetection](#)
  - [StartPersonTracking](#)
  - [StartSegmentDetection](#)
  - [StartTextDetection](#)

## Beispiele für die Route-53-Domainregistrierung mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Route-53-Domainregistrierung verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Erste Schritte

#### Hallo Route-53-Domainregistrierung

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit der Route-53-Domainregistrierung.

#### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.route53domains.Route53DomainsClient;
import software.amazon.awssdk.services.route53.model.Route53Exception;
import software.amazon.awssdk.services.route53domains.model.DomainPrice;
import software.amazon.awssdk.services.route53domains.model.ListPricesRequest;
import software.amazon.awssdk.services.route53domains.model.ListPricesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* This Java code examples performs the following operation:
*
* 1. Invokes ListPrices for at least one domain type, such as the "com" type
* and displays the prices for Registration and Renewal.
*
*/
public class HelloRoute53 {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <hostedZoneId> \n\n" +
            "Where:\n" +
            "    hostedZoneId - The id value of an existing hosted zone. \n";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainType = args[0];
        Region region = Region.US_EAST_1;
        Route53DomainsClient route53DomainsClient = Route53DomainsClient.builder()
            .region(region)
            .build();

        System.out.println(DASHES);
        System.out.println("Invokes ListPrices for at least one domain type.");
        listPrices(route53DomainsClient, domainType);
        System.out.println(DASHES);
    }

    public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
        try {
            ListPricesRequest pricesRequest = ListPricesRequest.builder()
                .maxItems(10)
                .tld(domainType)
                .build();
```

```
        ListPricesResponse response =
route53DomainsClient.listPrices(pricesRequest);
        List<DomainPrice> prices = response.prices();
        for (DomainPrice pr : prices) {
            System.out.println("Name: " + pr.name());
            System.out.println(
                "Registration: " + pr.registrationPrice().price() + " " +
pr.registrationPrice().currency());
            System.out.println("Renewal: " + pr.renewalPrice().price() + " " +
pr.renewalPrice().currency());
            System.out.println("Transfer: " + pr.transferPrice().price() + " " +
pr.transferPrice().currency());
            System.out.println("Transfer: " + pr.transferPrice().price() + " " +
pr.transferPrice().currency());
            System.out.println("Change Ownership: " +
pr.changeOwnershipPrice().price() + " "
                + pr.changeOwnershipPrice().currency());
            System.out.println(
                "Restoration: " + pr.restorationPrice().price() + " " +
pr.restorationPrice().currency());
            System.out.println(" ");
        }

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [ListPrices](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Themen

- [Aktionen](#)
- [Szenarien](#)

## Aktionen

### Überprüfen der Domainverfügbarkeit

Das folgende Codebeispiel zeigt, wie Sie die Verfügbarkeit einer Domain überprüfen.

SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void checkDomainAvailability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
            .domainName(domainSuggestion)
            .build();

        CheckDomainAvailabilityResponse response = route53DomainsClient
            .checkDomainAvailability(availabilityRequest);
        System.out.println(domainSuggestion + " is " +
response.availability().toString());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [CheckDomainAvailability](#) in der APIAWS SDK for Java 2.x -Referenz für .

### Überprüfen der Domainübertragbarkeit

Das folgende Codebeispiel zeigt, wie Sie die Übertragbarkeit einer Domain überprüfen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
            .domainName(domainSuggestion)
            .build();

        CheckDomainTransferabilityResponse response = route53DomainsClient
            .checkDomainTransferability(transferabilityRequest);
        System.out.println("Transferability: " +
response.transferability().transferable().toString());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [CheckDomainTransferability](#) in der APIAWS SDK for Java 2.x -Referenz für .

### Abrufen von Domainedetails

Das folgende Codebeispiel zeigt, wie Sie die Details für eine Domäne abrufen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void getDomainDetails(Route53DomainsClient route53DomainsClient,
String domainSuggestion) {
    try {
        GetDomainDetailRequest detailRequest = GetDomainDetailRequest.builder()
            .domainName(domainSuggestion)
            .build();

        GetDomainDetailResponse response =
route53DomainsClient.getDomainDetail(detailRequest);
        System.out.println("The contact first name is " +
response.registrantContact().firstName());
        System.out.println("The contact last name is " +
response.registrantContact().lastName());
        System.out.println("The contact org name is " +
response.registrantContact().organizationName());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [GetDomainDetail](#) in der APIAWS SDK for Java 2.x - Referenz für .

### Abrufen von Vorgangsdetails

Das folgende Codebeispiel zeigt, wie Sie Details zu einer -Operation abrufen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void getOperationalDetail(Route53DomainsClient
route53DomainsClient, String operationId) {
    try {
        GetOperationDetailRequest detailRequest =
        GetOperationDetailRequest.builder()
            .operationId(operationId)
            .build();

        GetOperationDetailResponse response =
        route53DomainsClient.getOperationDetail(detailRequest);
        System.out.println("Operation detail message is " + response.message());


    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [GetOperationDetail](#) in der APIAWS SDK for Java 2.x -Referenz zu .

## Abrufen vorgeschlagener Domainnamen

Das folgende Codebeispiel zeigt, wie Sie Vorschläge für Domainnamen abrufen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.



```
public static void listDomainSuggestions(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        GetDomainSuggestionsRequest suggestionsRequest =
        GetDomainSuggestionsRequest.builder()
            .domainName(domainSuggestion)
            .suggestionCount(5)
            .onlyAvailable(true)
            .build();

        GetDomainSuggestionsResponse response =
        route53DomainsClient.getDomainSuggestions(suggestionsRequest);
        List<DomainSuggestion> suggestions = response.suggestionsList();
        for (DomainSuggestion suggestion : suggestions) {
            System.out.println("Suggestion Name: " + suggestion.domainName());
            System.out.println("Availability: " + suggestion.availability());
            System.out.println(" ");
        }

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [GetDomainSuggestions](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Auflisten von Domainpreisen

Das folgende Codebeispiel zeigt, wie Sie Domain-Preise auflisten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
    try {
        ListPricesRequest pricesRequest = ListPricesRequest.builder()
            .tld(domainType)
            .build();

        ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
        listRes.stream()
            .flatMap(r -> r.prices().stream())
            .forEach(content -> System.out.println(" Name: " +
content.name() +
                " Registration: " + content.registrationPrice().price()
+ " "
                + content.registrationPrice().currency() +
                " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [ListPrices](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Domains

Das folgende Codebeispiel zeigt, wie Sie die registrierten Domänen auflisten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void listDomains(Route53DomainsClient route53DomainsClient) {
    try {
        ListDomainsIterable listRes =
route53DomainsClient.listDomainsPaginator();
        listRes.stream()
            .flatMap(r -> r.domains().stream())
            .forEach(content -> System.out.println("The domain name is " +
content.domainName()));
    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [ListDomains](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Listenoperationen

Das folgende Codebeispiel zeigt, wie Sie -Operationen auflisten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void listOperations(Route53DomainsClient route53DomainsClient) {
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        localDateTime = localDateTime.minusYears(1);
        Instant myTime = localDateTime.toInstant(zoneOffset);
    }
}
```

```

        ListOperationsRequest operationsRequest =
ListOperationsRequest.builder()
        .submittedSince(myTime)
        .build();

        ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
        listRes.stream()
        .flatMap(r -> r.operations().stream())
        .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                " Status: " + content.statusAsString() +
                " Date: " + content.submittedDate()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

```

- Weitere API-Informationen finden Sie unter [ListOperations](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Registrieren einer Domäne

Das folgende Codebeispiel zeigt, wie Sie eine Domain registrieren.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
        String domainSuggestion,
        String phoneNumber,
        String email,
        String firstName,

```

```
        String lastName,
        String city) {

    try {
        ContactDetail contactDetail = ContactDetail.builder()
            .contactType(ContactType.COMPANY)
            .state("LA")
            .countryCode(CountryCode.IN)
            .email(email)
            .firstName(firstName)
            .lastName(lastName)
            .city(city)
            .phoneNumber(phoneNumber)
            .organizationName("My Org")
            .addressLine1("My Address")
            .zipCode("123 123")
            .build();

        RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
            .adminContact(contactDetail)
            .registrantContact(contactDetail)
            .techContact(contactDetail)
            .domainName(domainSuggestion)
            .autoRenew(true)
            .durationInYears(1)
            .build();

        RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
        System.out.println("Registration requested. Operation Id: " +
response.operationId());
        return response.operationId();

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- Weitere API-Informationen finden Sie unter [RegisterDomain](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Anzeigen der Abrechnung

Das folgende Codebeispiel zeigt, wie Sie Fakturierungsdatensätze anzeigen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void listBillingRecords(Route53DomainsClient route53DomainsClient)
{
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        LocalDateTime localDateTime2 = localDateTime.minusYears(1);
        Instant myStartTime = localDateTime2.toInstant(zoneOffset);
        Instant myEndTime = localDateTime.toInstant(zoneOffset);

        ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
            .start(myStartTime)
            .end(myEndTime)
            .build();

        ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
        listRes.stream()
            .flatMap(r -> r.billingRecords().stream())
            .forEach(content -> System.out.println(" Bill Date:: " +
content.billDate() +
                " Operation: " + content.operationAsString() +
                " Price: " + content.price()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [ViewBilling](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Szenarien

### Erste Schritte mit Domains

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 Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example uses pagination methods where applicable. For example, to list
 * domains, the
 * listDomainsPaginator method is used. For more information about pagination,
```

```

* see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/pagination.html
*
* This Java code example performs the following operations:
*
* 1. List current domains.
* 2. List operations in the past year.
* 3. View billing for the account in the past year.
* 4. View prices for domain types.
* 5. Get domain suggestions.
* 6. Check domain availability.
* 7. Check domain transferability.
* 8. Request a domain registration.
* 9. Get operation details.
* 10. Optionally, get domain details.
*/

```

```

public class Route53Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <domainType> <phoneNumber> <email> <domainSuggestion>
<firstName> <lastName> <city>

            Where:
                domainType - The domain type (for example, com).\s
                phoneNumber - The phone number to use (for example,
+91.9966564xxx)      email - The email address to use.      domainSuggestion - The
domain suggestion (for example, findmy.accountants).\s
                firstName - The first name to use to register a domain.\s
                lastName - The last name to use to register a domain.\s
                city - the city to use to register a domain.\s
                """;

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainType = args[0];

```



```
String phoneNumber = args[1];
String email = args[2];
String domainSuggestion = args[3];
String firstName = args[4];
String lastName = args[5];
String city = args[6];
Region region = Region.US_EAST_1;
Route53DomainsClient route53DomainsClient = Route53DomainsClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon Route 53 domains example
scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. List current domains.");
listDomains(route53DomainsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. List operations in the past year.");
listOperations(route53DomainsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. View billing for the account in the past year.");
listBillingRecords(route53DomainsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. View prices for domain types.");
listPrices(route53DomainsClient, domainType);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Get domain suggestions.");
listDomainSuggestions(route53DomainsClient, domainSuggestion);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Check domain availability.");
checkDomainAvailability(route53DomainsClient, domainSuggestion);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Check domain transferability.");
        checkDomainTransferability(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Request a domain registration.");
        String opId = requestDomainRegistration(route53DomainsClient,
        domainSuggestion, phoneNumber, email, firstName,
            lastName, city);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get operation details.");
        getOperationalDetail(route53DomainsClient, opId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get domain details.");
        System.out.println("Note: You must have a registered domain to get
        details.");
        System.out.println("Otherwise, an exception is thrown that states ");
        System.out.println("Domain xxxxxxxx not found in xxxxxxxx account.");
        getDomainDetails(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);
    }

    public static void getDomainDetails(Route53DomainsClient route53DomainsClient,
    String domainSuggestion) {
        try {
            GetDomainDetailRequest detailRequest = GetDomainDetailRequest.builder()
                .domainName(domainSuggestion)
                .build();

            GetDomainDetailResponse response =
            route53DomainsClient.getDomainDetail(detailRequest);
            System.out.println("The contact first name is " +
            response.registrantContact().firstName());
            System.out.println("The contact last name is " +
            response.registrantContact().lastName());
            System.out.println("The contact org name is " +
            response.registrantContact().organizationName());
```

```
    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getOperationalDetail(Route53DomainsClient
route53DomainsClient, String operationId) {
    try {
        GetOperationDetailRequest detailRequest =
GetOperationDetailRequest.builder()
            .operationId(operationId)
            .build();

        GetOperationDetailResponse response =
route53DomainsClient.getOperationalDetail(detailRequest);
        System.out.println("Operation detail message is " + response.message());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
        String domainSuggestion,
        String phoneNumber,
        String email,
        String firstName,
        String lastName,
        String city) {

    try {
        ContactDetail contactDetail = ContactDetail.builder()
            .contactType(ContactType.COMPANY)
            .state("LA")
            .countryCode(CountryCode.IN)
            .email(email)
            .firstName(firstName)
            .lastName(lastName)
            .city(city)
            .phoneNumber(phoneNumber)
```

```
        .organizationName("My Org")
        .addressLine1("My Address")
        .zipCode("123 123")
        .build();

    RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
        .adminContact(contactDetail)
        .registrantContact(contactDetail)
        .techContact(contactDetail)
        .domainName(domainSuggestion)
        .autoRenew(true)
        .durationInYears(1)
        .build();

    RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
    System.out.println("Registration requested. Operation Id: " +
response.operationId());
    return response.operationId();

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
            .domainName(domainSuggestion)
            .build();

        CheckDomainTransferabilityResponse response = route53DomainsClient
            .checkDomainTransferability(transferabilityRequest);
        System.out.println("Transferability: " +
response.transferability().transferable().toString());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}

    public static void checkDomainAvailability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
                .domainName(domainSuggestion)
                .build();

            CheckDomainAvailabilityResponse response = route53DomainsClient
                .checkDomainAvailability(availabilityRequest);
            System.out.println(domainSuggestion + " is " +
response.availability().toString());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listDomainSuggestions(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            GetDomainSuggestionsRequest suggestionsRequest =
GetDomainSuggestionsRequest.builder()
                .domainName(domainSuggestion)
                .suggestionCount(5)
                .onlyAvailable(true)
                .build();

            GetDomainSuggestionsResponse response =
route53DomainsClient.getDomainSuggestions(suggestionsRequest);
            List<DomainSuggestion> suggestions = response.suggestionsList();
            for (DomainSuggestion suggestion : suggestions) {
                System.out.println("Suggestion Name: " + suggestion.domainName());
                System.out.println("Availability: " + suggestion.availability());
                System.out.println(" ");
            }

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

```
}

    public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
        try {
            ListPricesRequest pricesRequest = ListPricesRequest.builder()
                .tld(domainType)
                .build();

            ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
            listRes.stream()
                .flatMap(r -> r.prices().stream())
                .forEach(content -> System.out.println(" Name: " +
content.name() +
                    " Registration: " + content.registrationPrice().price()
+ " "
                    + content.registrationPrice().currency() +
                    " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listBillingRecords(Route53DomainsClient route53DomainsClient)
{
        try {
            Date currentDate = new Date();
            LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
            ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
            LocalDateTime localDateTime2 = localDateTime.minusYears(1);
            Instant myStartTime = localDateTime2.toInstant(zoneOffset);
            Instant myEndTime = localDateTime.toInstant(zoneOffset);

            ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
                .start(myStartTime)
                .end(myEndTime)
                .build();
```

```

        ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
        listRes.stream()
            .flatMap(r -> r.billingRecords().stream())
            .forEach(content -> System.out.println(" Bill Date:: " +
content.billDate() +
                " Operation: " + content.operationAsString() +
                " Price: " + content.price()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listOperations(Route53DomainsClient route53DomainsClient) {
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        localDateTime = localDateTime.minusYears(1);
        Instant myTime = localDateTime.toInstant(zoneOffset);

        ListOperationsRequest operationsRequest =
ListOperationsRequest.builder()
            .submittedSince(myTime)
            .build();

        ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
        listRes.stream()
            .flatMap(r -> r.operations().stream())
            .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                " Status: " + content.statusAsString() +
                " Date: " + content.submittedDate()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}

```

```
public static void listDomains(Route53DomainsClient route53DomainsClient) {
    try {
        ListDomainsIterable listRes =
route53DomainsClient.listDomainsPaginator();
        listRes.stream()
            .flatMap(r -> r.domains().stream())
            .forEach(content -> System.out.println("The domain name is " +
content.domainName()));
    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [CheckDomainAvailability](#)
  - [CheckDomainTransferability](#)
  - [GetDomainDetail](#)
  - [GetDomainSuggestions](#)
  - [GetOperationDetail](#)
  - [ListDomains](#)
  - [ListOperations](#)
  - [ListPrices](#)
  - [RegisterDomain](#)
  - [ViewBilling](#)

## Amazon S3-Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon S3 verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.



Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

## Erste Schritte

### Hello Amazon S3

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon S3.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBuckets(s3);
    }
}
```

```
    }

    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [ListBuckets](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Themen

- [Aktionen](#)
- [Szenarien](#)
- [Serverless-Beispiele](#)

## Aktionen

### CORS-Regeln einem Bucket hinzufügen

Das folgende Codebeispiel zeigt, wie CORS-Regeln (Cross-Origin Resource Sharing) zu einem S3-Bucket hinzugefügt werden.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import java.util.ArrayList;
import java.util.List;
import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.CORSRule;
import software.amazon.awssdk.services.s3.model.CORSConfiguration;
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3Cors {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <accountId>\s

                Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                accountId - The id of the account that owns the Amazon S3
bucket.

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
```

```
        .build();

        setCorsInformation(s3, bucketName, accountId);
        getBucketCorsInformation(s3, bucketName, accountId);
        deleteBucketCorsInformation(s3, bucketName, accountId);
        s3.close();
    }

    public static void deleteBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
        try {
            DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            s3.deleteBucketCors(bucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
        try {
            GetBucketCorsRequest bucketCorsRequest = GetBucketCorsRequest.builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            GetBucketCorsResponse corsResponse =
s3.getBucketCors(bucketCorsRequest);
            List<CORSRule> corsRules = corsResponse.getCorsRules();
            for (CORSRule rule : corsRules) {
                System.out.println("allowOrigins: " + rule.allowedOrigins());
                System.out.println("AllowedMethod: " + rule.allowedMethods());
            }

        } catch (S3Exception e) {

            System.err.println(e.awsErrorDetails().errorMessage());
```

```
        System.exit(1);
    }
}

public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {
    List<String> allowMethods = new ArrayList<>();
    allowMethods.add("PUT");
    allowMethods.add("POST");
    allowMethods.add("DELETE");

    List<String> allowOrigins = new ArrayList<>();
    allowOrigins.add("http://example.com");
    try {
        // Define CORS rules.
        CORSRule corsRule = CORSRule.builder()
            .allowedMethods(allowMethods)
            .allowedOrigins(allowOrigins)
            .build();

        List<CORSRule> corsRules = new ArrayList<>();
        corsRules.add(corsRule);
        CORSConfiguration configuration = CORSConfiguration.builder()
            .corsRules(corsRules)
            .build();

        PutBucketCorsRequest putBucketCorsRequest =
PutBucketCorsRequest.builder()
            .bucket(bucketName)
            .corsConfiguration(configuration)
            .expectedBucketOwner(accountId)
            .build();

        s3.putBucketCors(putBucketCorsRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [PutBucketCors](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Einem Bucket eine Lebenszyklus-Konfiguration hinzufügen

Das folgende Codebeispiel zeigt, wie Sie einem S3-Bucket eine Lebenszykluskonfiguration hinzufügen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.Transition;
import
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
import
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */
```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/

public class LifecycleConfiguration {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <accountId>\s

            Where:
                bucketName - The Amazon Simple Storage Service
(Amazon S3) bucket to upload an object into.
                accountId - The id of the account that owns the
Amazon S3 bucket.

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setLifecycleConfig(s3, bucketName, accountId);
        getLifecycleConfig(s3, bucketName, accountId);
        deleteLifecycleConfig(s3, bucketName, accountId);
        System.out.println("You have successfully created, updated, and
deleted a Lifecycle configuration");
        s3.close();
    }

    public static void setLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            // Create a rule to archive objects with the
"glacierobjects/" prefix to Amazon
            // S3 Glacier.

```

```
        LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
                        .prefix("glacierobjects/")
                        .build();

        Transition transition = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
                .days(0)
                .build();

        LifecycleRule rule1 = LifecycleRule.builder()
                .id("Archive immediately rule")
                .filter(ruleFilter)
                .transitions(transition)
                .status(ExpirationStatus.ENABLED)
                .build();

        // Create a second rule.
        Transition transition2 = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
                .days(0)
                .build();

        List<Transition> transitionList = new ArrayList<>();
        transitionList.add(transition2);

        LifecycleRuleFilter ruleFilter2 =
LifecycleRuleFilter.builder()
                        .prefix("glacierobjects/")
                        .build();

        LifecycleRule rule2 = LifecycleRule.builder()
                .id("Archive and then delete rule")
                .filter(ruleFilter2)
                .transitions(transitionList)
                .status(ExpirationStatus.ENABLED)
                .build();

        // Add the LifecycleRule objects to an ArrayList.
        ArrayList<LifecycleRule> ruleList = new ArrayList<>();
        ruleList.add(rule1);
        ruleList.add(rule2);
```



```
        BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                                .rules(ruleList)
                                .build();

        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                                .builder()
                                .bucket(bucketName)

.lifecycleConfiguration(lifecycleConfiguration)
                                .expectedBucketOwner(accountId)
                                .build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Retrieve the configuration and add a new rule.
    public static void getLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest = GetBucketLifecycleConfigurationRequest
                                .builder()
                                .bucket(bucketName)
                                .expectedBucketOwner(accountId)
                                .build();

            GetBucketLifecycleConfigurationResponse response = s3

.getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
            List<LifecycleRule> newList = new ArrayList<>();
            List<LifecycleRule> rules = response.rules();
            for (LifecycleRule rule : rules) {
                newList.add(rule);
            }
        }
    }
}
```

```
        // Add a new rule with both a prefix predicate and a tag
        predicate.
            LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
                .prefix("YearlyDocuments/")
                .build();

            Transition transition = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
                .days(3650)
                .build();

            LifecycleRule rule1 = LifecycleRule.builder()
                .id("NewRule")
                .filter(ruleFilter)
                .transitions(transition)
                .status(ExpirationStatus.ENABLED)
                .build();

            // Add the new rule to the list.
            newList.add(rule1);
            BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                .rules(newList)
                .build();

            PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                .builder()
                .bucket(bucketName)

.lifecycleConfiguration(lifecycleConfiguration)
                .expectedBucketOwner(accountId)
                .build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
// Delete the configuration from the Amazon S3 bucket.
public static void deleteLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
    try {
        DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
DeleteBucketLifecycleRequest
                                .builder()
                                .bucket(bucketName)
                                .expectedBucketOwner(accountId)
                                .build();

        s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);


    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [PutBucketLifecycleConfiguration](#) in der APIAWS SDK for Java 2.x -Referenz für .

Einem Bucket eine Richtlinie hinzufügen

Das folgende Codebeispiel zeigt, wie Sie einem S3-Bucket eine Richtlinie hinzufügen.

SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import java.io.IOException;
```

```
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetBucketPolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <polFile>

            Where:
                bucketName - The Amazon S3 bucket to set the policy on.
                polFile - A JSON file containing the policy (see the Amazon S3
Readme for an example).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String polFile = args[1];
        String policyText = getBucketPolicyFromFile(polFile);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setPolicy(s3, bucketName, policyText);
        s3.close();
    }
}
```

```
public static void setPolicy(S3Client s3, String bucketName, String policyText)
{
    System.out.println("Setting policy:");
    System.out.println("----");
    System.out.println(policyText);
    System.out.println("----");
    System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

    try {
        PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
            .bucket(bucketName)
            .policy(policyText)
            .build();

        s3.putBucketPolicy(policyReq);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    System.out.println("Done!");
}

// Loads a JSON-formatted policy from a file
public static String getBucketPolicyFromFile(String policyFile) {

    StringBuilder fileText = new StringBuilder();
    try {
        List<String> lines = Files.readAllLines(Paths.get(policyFile),
StandardCharsets.UTF_8);
        for (String line : lines) {
            fileText.append(line);
        }

    } catch (IOException e) {
        System.out.format("Problem reading file: \"%s\"", policyFile);
        System.out.println(e.getMessage());
    }

    try {
        final JsonParser parser = new
ObjectMapper().getFactory().createParser(fileText.toString());
```

```
        while (parser.nextToken() != null) {
        }


    } catch (IOException jpe) {
        jpe.printStackTrace();
    }
    return fileText.toString();
}
}
```

- Weitere API-Informationen finden Sie unter [PutBucketPolicy](#) in der APIAWS SDK for Java 2.x - Referenz für .

Ein Objekt von einem Bucket in einen anderen Bucket kopieren

Im folgenden Codebeispiel wird demonstriert, wie Sie ein S3-Objekt von einem Bucket in einen anderen kopieren.

SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Kopieren eines Objekts mit einem [S3Client](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
*/

public class CopyObject {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <objectKey> <fromBucket> <toBucket>

            Where:
                objectKey - The name of the object (for example, book.pdf).
                fromBucket - The S3 bucket name that contains the object (for
example, bucket1).
                toBucket - The S3 bucket to copy the object to (for example,
bucket2).

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String objectKey = args[0];
        String fromBucket = args[1];
        String toBucket = args[2];
        System.out.format("Copying object %s from bucket %s to %s\n", objectKey,
fromBucket, toBucket);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        copyBucketObject(s3, fromBucket, objectKey, toBucket);
        s3.close();
    }

    public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(fromBucket)
            .sourceKey(objectKey)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
            .build();
    }
}
```

```
    try {
        CopyObjectResponse copyRes = s3.copyObject(copyReq);
        return copyRes.copyObjectResult().toString();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

Verwenden Sie einen [S3TransferManager](#), um [ein Objekt von einem Bucket in einen anderen zu kopieren](#). Sehen Sie sich die [vollständige Datei](#) an und [testen](#) Sie sie.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;

public String copyObject(S3TransferManager transferManager, String bucketName,
    String key, String destinationBucket, String destinationKey) {
    CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
        .sourceBucket(bucketName)
        .sourceKey(key)
        .destinationBucket(destinationBucket)
        .destinationKey(destinationKey)
        .build();

    CopyRequest copyRequest = CopyRequest.builder()
        .copyObjectRequest(copyObjectRequest)
        .build();

    Copy copy = transferManager.copy(copyRequest);
}
```



```

    CompletedCopy completedCopy = copy.completionFuture().join();
    return completedCopy.response().copyObjectResult().eTag();
}

```

- Weitere API-Informationen finden Sie unter [CopyObject](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen eines -Buckets

Das folgende Codebeispiel zeigen, wie Sie einen S3 Bucket erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CreateBucket {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = ""

```

```

        Usage:
            <bucketName>\s

        Where:
            bucketName - The name of the bucket to create. The bucket name
must be unique, or an error occurs.
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    System.out.format("Creating a bucket named %s\n", bucketName);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    createBucket(s3, bucketName);
    s3.close();
}

public static void createBucket(S3Client s3Client, String bucketName) {
    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {

```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [CreateBucket](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Eine Richtlinie aus einem Bucket löschen

Das folgende Codebeispiel zeigt, wie Sie eine Richtlinie aus einem S3-Bucket löschen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteBucketPolicy {
    public static void main(String[] args) {

        final String usage = ""
```

```

        Usage:
            <bucketName>

        Where:
            bucketName - The Amazon S3 bucket to delete the policy from (for
example, bucket1).""";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteS3BucketPolicy(s3, bucketName);
        s3.close();
    }

    // Delete the bucket policy.
    public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {
        DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
            .bucket(bucketName)
            .build();

        try {
            s3.deleteBucketPolicy(delReq);
            System.out.println("Done!");
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- Weitere API-Informationen finden Sie unter [DeleteBucketPolicy](#) in der APIAWS SDK for Java 2.x -Referenz zu .

## Einen leeren Bucket löschen

Im folgenden Codebeispiel wird demonstriert, wie Sie einen leeren S3 Bucket löschen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucket)
    .build();

s3.deleteBucket(deleteBucketRequest);
s3.close();
```

- Weitere API-Informationen finden Sie unter [DeleteBucket](#) in der APIAWS SDK for Java 2.x - Referenz zu .

## Mehrere Objekte löschen

Im folgenden Codebeispiel wird demonstriert, wie Sie mehrere Objekte aus einem S3 Bucket löschen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
```

```
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteMultiObjects {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <bucketName>

                Where:
                    bucketName - the Amazon S3 bucket name.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void deleteBucketObjects(S3Client s3, String bucketName) {
        // Upload three sample objects to the specified Amazon S3 bucket.
        ArrayList<ObjectIdentifier> keys = new ArrayList<>();
        PutObjectRequest putObj;
        ObjectIdentifier objectId;
```

```
for (int i = 0; i < 3; i++) {
    String keyName = "delete object example " + i;
    ObjectIdentifier objectId = ObjectIdentifier.builder()
        .key(keyName)
        .build();

    PutObjectRequest putOb = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(keyName)
        .build();

    s3.putObject(putOb, RequestBody.fromString(keyName));
    keys.add(objectId);
}

System.out.println(keys.size() + " objects successfully created.");

// Delete multiple objects in one request.
Delete del = Delete.builder()
    .objects(keys)
    .build();

try {
    DeleteObjectsRequest multiObjectDeleteRequest =
DeleteObjectsRequest.builder()
    .bucket(bucketName)
    .delete(del)
    .build();

    s3.deleteObjects(multiObjectDeleteRequest);
    System.out.println("Multiple objects are deleted!");

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [DeleteObjects](#) in der APIAWS SDK for Java 2.x - Referenz zu .

## Die Website-Konfiguration eines Buckets löschen

Das folgende Codebeispiel zeigt, wie Sie die Website-Konfiguration aus einem S3-Bucket löschen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = ""

                Usage:      <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to delete the website
configuration from.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
```



```
        System.out.format("Deleting website configuration for Amazon S3 bucket: %s\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketWebsiteConfig(s3, bucketName);
        System.out.println("Done!");
        s3.close();
    }

    public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName) {
        DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
            .bucket(bucketName)
            .build();

        try {
            s3.deleteBucketWebsite(delReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.out.println("Failed to delete website configuration!");
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteBucketWebsite](#) in der APIAWS SDK for Java 2.x -Referenz zu .

## Das Vorhandensein und den Inhaltstyp eines Objekts bestimmen

Das folgende Codebeispiel zeigt, wie Sie das Vorhandensein und den Inhaltstyp eines Objekts in einem S3-Bucket bestimmen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Bestimmen Sie den Inhaltstyp eines Objekts.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectContentType {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <keyName>>

                Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
```

```

        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getContentType(s3, bucketName, keyName);
        s3.close();
    }

    public static void getContentType(S3Client s3, String bucketName, String
keyName) {
        try {
            HeadObjectRequest objectRequest = HeadObjectRequest.builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            HeadObjectResponse objectHead = s3.headObject(objectRequest);
            String type = objectHead.contentType();
            System.out.println("The object content type is " + type);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

Rufen Sie den Wiederherstellungsstatus eines Objekts ab.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

public class GetObjectRestoreStatus {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <keyName>\s
    }
}

```

```

        Where:
            bucketName - The Amazon S3 bucket name.\s
            keyName - A key name that represents the object.\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String keyName = args[1];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    checkStatus(s3, bucketName, keyName);
    s3.close();
}

public static void checkStatus(S3Client s3, String bucketName, String keyName) {
    try {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        HeadObjectResponse response = s3.headObject(headObjectRequest);
        System.out.println("The Amazon S3 object restoration status is " +
response.restore());

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}
}

```

- Weitere API-Informationen finden Sie unter [HeadObject](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Herunterladen von Objekten in ein lokales Verzeichnis

Das folgende Codebeispiel zeigt, wie Sie alle Objekte aus einem Amazon Simple Storage Service (Amazon S3)-Bucket in ein lokales Verzeichnis herunterladen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Verwenden Sie einen [S3TransferManager](#), um [alle S3-Objekte in denselben S3-Bucket herunterzuladen](#). S3 Sehen Sie sich die [vollständige Datei](#) an und [testen](#) Sie sie.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;

import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;

    public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
        URI destinationPathURI, String bucketName) {
        DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
            .destination(Paths.get(destinationPathURI))
            .bucket(bucketName)
```

```

        .build());
    CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryDownload.failedTransfers().size();
}

```

- Weitere API-Informationen finden Sie unter [DownloadDirectory](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Aktivieren von Benachrichtigungen

Das folgende Codebeispiel zeigt, wie Sie Benachrichtigungen für einen S3-Bucket aktivieren.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Event;
import software.amazon.awssdk.services.s3.model.NotificationConfiguration;
import
    software.amazon.awssdk.services.s3.model.PutBucketNotificationConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.TopicConfiguration;
import java.util.ArrayList;
import java.util.List;

public class SetBucketEventBridgeNotification {
    public static void main(String[] args) {
        final String usage = ""

        Usage:

```

```

        <bucketName>\s

        Where:
            bucketName - The Amazon S3 bucket.\s
            topicArn - The Simple Notification Service topic ARN.\s
            id - An id value used for the topic configuration. This value is
displayed in the AWS Management Console.\s
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String topicArn = args[1];
    String id = args[2];
    Region region = Region.US_EAST_1;
    S3Client s3Client = S3Client.builder()
        .region(region)
        .build();

    setBucketNotification(s3Client, bucketName, topicArn, id);
    s3Client.close();
}

public static void setBucketNotification(S3Client s3Client, String bucketName,
String topicArn, String id) {
    try {
        List<Event> events = new ArrayList<>();
        events.add(Event.S3_OBJECT_CREATED_PUT);

        TopicConfiguration config = TopicConfiguration.builder()
            .topicArn(topicArn)
            .events(events)
            .id(id)
            .build();

        List<TopicConfiguration> topics = new ArrayList<>();
        topics.add(config);

        NotificationConfiguration configuration =
NotificationConfiguration.builder()
            .topicConfigurations(topics)

```

```
        .build();

        PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest
        .builder()
        .bucket(bucketName)
        .notificationConfiguration(configuration)
        .skipDestinationValidation(true)
        .build();

        // Set the bucket notification configuration.
        s3Client.putBucketNotificationConfiguration(configurationRequest);
        System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [PutBucketNotificationConfiguration](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Ein Objekt aus einem Bucket abrufen

Im folgenden Codebeispiel wird veranschaulicht, wie Sie Daten aus einem Objekt in einem S3 Bucket lesen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Lesen Sie Daten als Byte-Array mit einem [S3Client](#).

```
import software.amazon.awssdk.core.ResponseBytes;
```



```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectData {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName> <path>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                path - The path where the file is written to.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();
```

```
        getObjectBytes(s3, bucketName, keyName, path);
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Verwenden Sie ein [S3TransferManager](#), um [ein Objekt in einem S3-Bucket in eine lokale Datei herunterzuladen](#). S3 Sehen Sie sich die [vollständige Datei](#) an und [testen](#) Sie sie.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
```

```
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;

    public Long downloadFile(S3TransferManager transferManager, String bucketName,
                            String key, String downloadedFilePath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
            .getObjectRequest(b -> b.bucket(bucketName).key(key))
            .addTransferListener(LoggingTransferListener.create())
            .destination(Paths.get(downloadedFilePath))
            .build();

        FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
downloadResult.response().contentLength());
        return downloadResult.response().contentLength();
    }
```

Lesen Sie Tags, die zu einem Objekt gehören, mit einem [S3Client](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tag;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/

public class GetObjectTags {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listTags(s3, bucketName, keyName);
        s3.close();
    }

    public static void listTags(S3Client s3, String bucketName, String keyName) {
        try {
            GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            GetObjectTaggingResponse tags = s3.getObjectTagging(getTaggingRequest);
            List<Tag> tagSet = tags.tagSet();
            for (Tag tag : tagSet) {
                System.out.println(tag.key());
            }
        }
    }
}
```

```

        System.out.println(tag.value());
    }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

Rufen Sie eine URL für ein Objekt mit einem [S3Client](#) ab.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetUrlRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.net.URL;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectUrl {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <keyName>\s

                Where:
                bucketName - The Amazon S3 bucket name.
                keyName - A key name that represents the object.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}

```

```
    }

    String bucketName = args[0];
    String keyName = args[1];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    getURL(s3, bucketName, keyName);
    s3.close();
}

public static void getURL(S3Client s3, String bucketName, String keyName) {
    try {
        GetUrlRequest request = GetUrlRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        URL url = s3.utilities().getUrl(request);
        System.out.println("The URL for " + keyName + " is " + url);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

Rufen Sie ein Objekt mithilfe des S3Presigner-Client-Objekts mit einem [S3Client](#) ab.

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
```

```
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents a text file.\s
            """;

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Presigner presigner = S3Presigner.builder()
            .region(region)
            .build();

        getPresignedUrl(presigner, bucketName, keyName);
        presigner.close();
    }

    public static void getPresignedUrl(S3Presigner presigner, String bucketName,
String keyName) {
        try {
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
```

```
        .key(keyName)
        .build();

    GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
        .signatureDuration(Duration.ofMinutes(60))
        .getObjectRequest(getObjectRequest)
        .build();

    PresignedGetObjectRequest presignedGetObjectRequest =
presigner.presignGetObject(getObjectPresignRequest);
    String theUrl = presignedGetObjectRequest.url().toString();
    System.out.println("Presigned URL: " + theUrl);
    HttpURLConnection connection = (HttpURLConnection)
presignedGetObjectRequest.url().openConnection();
    presignedGetObjectRequest.httpRequest().headers().forEach((header,
values) -> {
        values.forEach(value -> {
            connection.setRequestProperty(header, value);
        });
    });

    // Send any request payload that the service needs (not needed when
// isBrowserExecutable is true).
    if (presignedGetObjectRequest.signedPayload().isPresent()) {
        connection.setDoOutput(true);

        try (InputStream signedPayload =
presignedGetObjectRequest.signedPayload().get().asInputStream();
            OutputStream httpOutputStream =
connection.getOutputStream()) {
            IoUtils.copy(signedPayload, httpOutputStream);
        }
    }

    // Download the result of executing the request.
    try (InputStream content = connection.getInputStream()) {
        System.out.println("Service returned response: ");
        IoUtils.copy(content, System.out);
    }

} catch (S3Exception | IOException e) {
    e.printStackTrace();
}
```



```
}  
}
```

Rufen Sie ein Objekt mit einem ResponseTransformer Objekt und [S3Client](#) ab.

```
import software.amazon.awssdk.core.ResponseBytes;  
import software.amazon.awssdk.core.sync.ResponseTransformer;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.GetObjectRequest;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
import software.amazon.awssdk.services.s3.model.GetObjectResponse;  
import java.io.File;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.OutputStream;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
  
public class GetDataResponseTransformer {  
    public static void main(String[] args) {  
        final String usage = ""  
  
            Usage:  
            <bucketName> <keyName> <path>  
  
            Where:  
            bucketName - The Amazon S3 bucket name.\s  
            keyName - The key name.\s  
            path - The path where the file is written to.\s  
            "";  
  
        if (args.length != 3) {  
            System.out.println(usage);  
            System.exit(1);  
        }  
    }  
}
```

```
    }

    String bucketName = args[0];
    String keyName = args[1];
    String path = args[2];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    getObjectBytes(s3, bucketName, keyName, path);
    s3.close();
}

public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
    try {
        GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        ResponseBytes<GetObjectResponse> objectBytes =
s3.getObject(objectRequest, ResponseTransformer.toBytes());
        byte[] data = objectBytes.asByteArray();

        // Write the data to a local file.
        File myFile = new File(path);
        OutputStream os = new FileOutputStream(myFile);
        os.write(data);
        System.out.println("Successfully obtained bytes from an S3 object");
        os.close();

    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [GetObject](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Die ACL eines Buckets abrufen

Das folgende Codebeispiel zeigt, wie Sie die Zugriffssteuerungsliste (ACL) eines S3-Buckets abrufen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;
import software.amazon.awssdk.services.s3.model.Grant;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetAcl {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <objectKey>

                Where:
                bucketName - The Amazon S3 bucket to get the access control list
                (ACL) for.
```

```
        objectKey - The object to get the ACL for.\s
        """);

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String objectKey = args[1];
    System.out.println("Retrieving ACL for object: " + objectKey);
    System.out.println("in bucket: " + bucketName);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    getBucketACL(s3, objectKey, bucketName);
    s3.close();
    System.out.println("Done!");
}

public static String getBucketACL(S3Client s3, String objectKey, String
bucketName) {
    try {
        GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
        List<Grant> grants = aclRes.grants();
        String grantee = "";
        for (Grant grant : grants) {
            System.out.format("  %s: %s\n", grant.grantee().id(),
grant.permission());
            grantee = grant.grantee().id();
        }

        return grantee;
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```


```
        return "";  
    }  
}
```

- Weitere API-Informationen finden Sie unter [GetBucketAcl](#) in der APIAWS SDK for Java 2.x - Referenz für .

Die Richtlinie für einen Bucket abrufen

Das folgende Codebeispiel zeigt, wie Sie die Richtlinie für einen S3-Bucket abrufen.

SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.services.s3.model.S3Exception;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;  
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
  
public class GetBucketPolicy {  
    public static void main(String[] args) {  
        final String usage = ""  
  
            Usage:  
            <bucketName>
```

```
        Where:
            bucketName - The Amazon S3 bucket to get the policy from.
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    String polText = getPolicy(s3, bucketName);
    System.out.println("Policy Text: " + polText);
    s3.close();
}

public static String getPolicy(S3Client s3, String bucketName) {
    String policyText;
    System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
    GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
        policyText = policyRes.policy();
        return policyText;
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
}
```

- Weitere API-Informationen finden Sie unter [GetBucketPolicy](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Buckets auflisten

Das folgende Codebeispiel zeigt, wie Sie S3-Buckets auflisten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listAllBuckets(s3);
    }

    public static void listAllBuckets(S3Client s3) {
        ListBucketsResponse response = s3.listBuckets();
        List<Bucket> bucketList = response.buckets();
    }
}
```

```
        for (Bucket bucket: bucketList) {
            System.out.println("Bucket name "+bucket.name());
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [ListBuckets](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Laufende mehrteilige Uploads auflisten

Das folgende Codebeispiel veranschaulicht, wie Sie in Bearbeitung befindliche, mehrteilige Uploads in einen S3 Bucket auflisten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListMultipartUploads {
```



```
public static void main(String[] args) {
    final String usage = ""

        Usage:
            <bucketName>\s

        Where:
            bucketName - The name of the Amazon S3 bucket where an in-
progress multipart upload is occurring.
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();
    listUploads(s3, bucketName);
    s3.close();
}

public static void listUploads(S3Client s3, String bucketName) {
    try {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();

        ListMultipartUploadsResponse response =
s3.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();
        for (MultipartUpload upload : uploads) {
            System.out.println("Upload in progress: Key = \"\" + upload.key() +
\"\", id = \"\" + upload.uploadId());
        }

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}  
}
```

- Weitere API-Informationen finden Sie unter [ListMultipartUploads](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Objekte in einem Bucket auflisten

Im folgenden Codebeispiel wird veranschaulicht, wie Sie Objekte in einem S3 Bucket auflisten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;  
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
import software.amazon.awssdk.services.s3.model.S3Object;  
import java.util.List;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
  
public class ListObjects {  
    public static void main(String[] args) {  
        final String usage = ""  
  
            Usage:  
            <bucketName>\s
```

```
        Where:
            bucketName - The Amazon S3 bucket from which objects are read.\s
            """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    listBucketObjects(s3, bucketName);
    s3.close();
}

public static void listBucketObjects(S3Client s3, String bucketName) {
    try {
        ListObjectsRequest listObjects = ListObjectsRequest
            .builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.print("\n The name of the key is " + myValue.key());
            System.out.print("\n The object is " + calcKb(myValue.size()) + "
KBs");

            System.out.print("\n The owner is " + myValue.owner());
        }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// convert bytes to kbs.
private static long calcKb(Long val) {
```

```
        return val / 1024;
    }
}
```

Listet Objekte mithilfe der Paginierung auf.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;

public class ListObjectsPaginated {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName>\s

                Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsV2Request listReq = ListObjectsV2Request.builder()
                .bucket(bucketName)
```

```
        .maxKeys(1)
        .build();

    ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
    listRes.stream()
        .flatMap(r -> r.contents().stream())
        .forEach(content -> System.out.println(" Key: " + content.key()
+ " size = " + content.size()));

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [ListObjectsV2](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Eine archivierte Kopie eines Objekts wiederherstellen

Das folgende Codebeispiel zeigt, wie Sie eine archivierte Kopie eines Objekts in einem S3 Bucket wiederherstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.RestoreRequest;
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tier;
```

```
/*
 * For more information about restoring an object, see "Restoring an archived
 object" at
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html
 *
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RestoreObject {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                    <bucketName> <keyName> <expectedBucketOwner>

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - The key name of an object with a Storage class value
of Glacier.\s
                    expectedBucketOwner - The account that owns the bucket (you can
obtain this value from the AWS Management Console).\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String expectedBucketOwner = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
        s3.close();
    }
}
```

```
public static void restoreS3Object(S3Client s3, String bucketName, String
keyName, String expectedBucketOwner) {
    try {
        RestoreRequest restoreRequest = RestoreRequest.builder()
            .days(10)

        .glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
            .build();

        RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
            .expectedBucketOwner(expectedBucketOwner)
            .bucket(bucketName)
            .key(keyName)
            .restoreRequest(restoreRequest)
            .build();

        s3.restoreObject(objectRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [RestoreObject](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Eine neue ACL für einen Bucket festlegen

Das folgende Codebeispiel zeigt, wie Sie eine neue Zugriffssteuerungsliste (ACL) für einen S3-Bucket festlegen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import java.util.ArrayList;
import java.util.List;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
import software.amazon.awssdk.services.s3.model.Type;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetAcl {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <bucketName> <id>\s

            Where:
            bucketName - The Amazon S3 bucket to grant permissions on.\s
            id - The ID of the owner of this bucket (you can get this value
from the AWS Management Console).
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String id = args[1];
        System.out.format("Setting access \n");
        System.out.println(" in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
```



```
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

setBucketAcl(s3, bucketName, id);
System.out.println("Done!");
s3.close();
}

public static void setBucketAcl(S3Client s3, String bucketName, String id) {
    try {
        Grant ownerGrant = Grant.builder()
            .grantee(builder -> builder.id(id)
                .type(Type.CANONICAL_USER))
            .permission(Permission.FULL_CONTROL)
            .build();

        List<Grant> grantList2 = new ArrayList<>();
        grantList2.add(ownerGrant);

        AccessControlPolicy acl = AccessControlPolicy.builder()
            .owner(builder -> builder.id(id))
            .grants(grantList2)
            .build();

        PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
            .bucket(bucketName)
            .accessControlPolicy(acl)
            .build();

        s3.putBucketAcl(putAclReq);

    } catch (S3Exception e) {
        e.printStackTrace();
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [PutBucketAcl](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Die Website-Konfiguration für einen Bucket festlegen

Das folgende Codebeispiel zeigt, wie Sie die Website-Konfiguration für einen S3-Bucket festlegen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.IndexDocument;
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SetWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <bucketName> [indexdoc]\s

                Where:
                bucketName  - The Amazon S3 bucket to set the website
configuration on.\s
                indexdoc - The index document, ex. 'index.html'
                        If not specified, 'index.html' will be set.

                """;

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String bucketName = args[0];
    String indexDoc = "index.html";
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setWebsiteConfig(s3, bucketName, indexDoc);
    s3.close();
}

public static void setWebsiteConfig(S3Client s3, String bucketName, String
indexDoc) {
    try {
        WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()
            .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
            .build();

        PutBucketWebsiteRequest pubWebsiteReq =
PutBucketWebsiteRequest.builder()
            .bucket(bucketName)
            .websiteConfiguration(websiteConfig)
            .build();

        s3.putBucketWebsite(pubWebsiteReq);
        System.out.println("The call was successful");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [PutBucketWebsite](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Ein Objekt in einen Bucket hochladen

Im folgenden Codebeispiel wird veranschaulicht, wie Sie ein Objekt in einen S3 Bucket hochladen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Laden Sie eine Datei mit einem [S3Client](#) in einen Bucket hoch.

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class PutObject {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <objectKey> <objectPath>\s

                Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                objectKey - The object to upload (for example, book.pdf).
```

```
        objectPath - The path where the file is located (for example, C:/
AWS/book2.pdf).\s
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String objectKey = args[1];
    String objectPath = args[2];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    putS3Object(s3, bucketName, objectKey, objectPath);
    s3.close();
}

// This example uses RequestBody.fromFile to avoid loading the whole file into
// memory.
public static void putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {
    try {
        Map<String, String> metadata = new HashMap<>();
        metadata.put("x-amz-meta-myVal", "test");
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .metadata(metadata)
            .build();

        s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
        System.out.println("Successfully placed " + objectKey + " into bucket "
+ bucketName);

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

Verwenden Sie einen [S3TransferManager](#), um [eine Datei in einen Bucket hochzuladen](#). Sehen Sie sich die [vollständige Datei](#) an und [testen](#) Sie sie.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public String uploadFile(S3TransferManager transferManager, String bucketName,
                            String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .addTransferListener(LoggingTransferListener.create())
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }
```

Laden Sie ein Objekt mit einem [S3Client](#) in einen Bucket hoch und legen Sie Tags fest.

```
    public static void putS3ObjectTags(S3Client s3, String bucketName, String
    objectKey, String objectPath) {
        try {
            Tag tag1 = Tag.builder()
                .key("Tag 1")
                .value("This is tag 1")
                .build();
```

```
        Tag tag2 = Tag.builder()
            .key("Tag 2")
            .value("This is tag 2")
            .build();

        List<Tag> tags = new ArrayList<>();
        tags.add(tag1);
        tags.add(tag2);

        Tagging allTags = Tagging.builder()
            .tagSet(tags)
            .build();

        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .tagging(allTags)
            .build();

        s3.putObject(putOb, RequestBody.fromBytes(getObjectFile(objectPath)));

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {
    try {
        GetObjectTaggingRequest taggingRequest =
GetObjectTaggingRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
        List<Tag> obTags = getTaggingRes.tagSet();
        for (Tag sinTag : obTags) {
            System.out.println("The tag key is: " + sinTag.key());
            System.out.println("The tag value is: " + sinTag.value());
        }
    }
}
```

```
// Replace the object's tags with two new tags.
Tag tag3 = Tag.builder()
    .key("Tag 3")
    .value("This is tag 3")
    .build();

Tag tag4 = Tag.builder()
    .key("Tag 4")
    .value("This is tag 4")
    .build();

List<Tag> tags = new ArrayList<>();
tags.add(tag3);
tags.add(tag4);

Tagging updatedTags = Tagging.builder()
    .tagSet(tags)
    .build();

PutObjectTaggingRequest taggingRequest1 =
PutObjectTaggingRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .tagging(updatedTags)
    .build();

s3.putObjectTagging(taggingRequest1);
GetObjectTaggingResponse getTaggingRes2 =
s3.getObjectTagging(taggingRequest);
List<Tag> modTags = getTaggingRes2.tagSet();
for (Tag sinTag : modTags) {
    System.out.println("The tag key is: " + sinTag.key());
    System.out.println("The tag value is: " + sinTag.value());
}

} catch (S3Exception e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

// Return a byte array.
private static byte[] getObjectFile(String filePath) {
    FileInputStream fileInputStream = null;
```



```
byte[] byteArray = null;

try {
    File file = new File(filePath);
    byteArray = new byte[(int) file.length()];
    FileInputStream = new FileInputStream(file);
    FileInputStream.read(byteArray);

} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (fileInputStream != null) {
        try {
            fileInputStream.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

return byteArray;
}
```

Laden Sie ein Objekt mit einem [S3Client](#) in einen Bucket hoch und legen Sie Metadaten fest.

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html

```

```
*/
public class PutObjectMetadata {
    public static void main(String[] args) {
        final String USAGE = ""

            Usage:
                <bucketName> <objectKey> <objectPath>\s

            Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                objectKey - The object to upload (for example, book.pdf).
                objectPath - The path where the file is located (for example, C:/
AWS/book2.pdf).\s
                """;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        System.out.println("Putting object " + objectKey + " into bucket " +
bucketName);
        System.out.println(" in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }

    // This example uses RequestBody.fromFile to avoid loading the whole file into
    // memory.
    public static void putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("author", "Mary Doe");
            metadata.put("version", "1.0.0.0");
        }
    }
}
```

```

        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .metadata(metadata)
            .build();

        s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
        System.out.println("Successfully placed " + objectKey + " into bucket "
+ bucketName);

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

Laden Sie ein Objekt mit einem [S3Client](#) in einen Bucket hoch und legen Sie einen Wert für die Objektaufbewahrung fest.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class PutObjectRetention {
    public static void main(String[] args) {
        final String usage = ""

```

```
Usage:
    <key> <bucketName>\s

Where:
    key - The name of the object (for example, book.pdf).\s
    bucketName - The Amazon S3 bucket name that contains the object
(for example, bucket1).\s
    """";

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String key = args[0];
    String bucketName = args[1];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setRetentionPeriod(s3, key, bucketName);
    s3.close();
}

public static void setRetentionPeriod(S3Client s3, String key, String bucket) {
    try {
        LocalDate localDate = LocalDate.parse("2020-07-17");
        LocalDateTime localDateTime = localDate.atStartOfDay();
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

        ObjectLockRetention lockRetention = ObjectLockRetention.builder()
            .mode("COMPLIANCE")
            .retainUntilDate(instant)
            .build();

        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
            .bucket(bucket)
            .key(key)
            .bypassGovernanceRetention(true)
            .retention(lockRetention)
            .build();
```

```
object // To set Retention on an object, the Amazon S3 bucket must support
// locking, otherwise an exception is thrown.
s3.putObjectRetention(retentionRequest);
System.out.print("An object retention configuration was successfully
placed on the object");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [PutObject](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Hochladen eines Verzeichnisses in einen Bucket

Das folgende Codebeispiel zeigt, wie Sie ein lokales Verzeichnis rekursiv in einen Amazon Simple Storage Service (Amazon S3)-Bucket hochladen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Verwenden Sie einen [S3TransferManager](#), um [ein lokales Verzeichnis hochzuladen](#). Sehen Sie sich die [vollständige Datei](#) an und [testen](#) Sie sie.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
```

```
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public Integer uploadDirectory(S3TransferManager transferManager,
        URI sourceDirectory, String bucketName) {
        DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
        .source(Paths.get(sourceDirectory))
        .bucket(bucketName)
        .build());

        CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
        completedDirectoryUpload.failedTransfers()
            .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
}
```

- Weitere API-Informationen finden Sie unter [UploadDirectory](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Verwenden von SQL mit Amazon S3 Select

Das folgende Codebeispiel zeigt, wie eine Teilmenge von Daten mit SQL abgerufen wird.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Das folgende Beispiel zeigt eine Abfrage mit einem JSON-Objekt. Das [vollständige Beispiel](#) zeigt auch die Verwendung eines CSV-Objekts.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.CSVInput;
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.OutputSerialization;
import software.amazon.awssdk.services.s3.model.Progress;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
import software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

public class SelectObjectContentExample {
    static final Logger logger =
        LoggerFactory.getLogger(SelectObjectContentExample.class);
    static final String BUCKET_NAME = "select-object-content-" + UUID.randomUUID();
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.csv";
    static String URL_JSON = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.json";
```

```
public static void main(String[] args) {
    SelectObjectContentExample selectObjectContentExample = new
SelectObjectContentExample();
    try {
        SelectObjectContentExample.setUp();
        selectObjectContentExample.runSelectObjectContentMethodForJSON();
        selectObjectContentExample.runSelectObjectContentMethodForCSV();
    } catch (SdkException e) {
        logger.error(e.getMessage(), e);
        System.exit(1);
    } finally {
        SelectObjectContentExample.tearDown();
    }
}

EventStreamInfo runSelectObjectContentMethodForJSON() {
    // Set up request parameters.
    final String queryExpression = "select * from s3object[*][*] c where c.area
< 350000";
    final String fileType = FILE_JSON;

    InputSerialization inputSerialization = InputSerialization.builder()
        .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
        .compressionType(CompressionType.NONE)
        .build();

    OutputSerialization outputSerialization = OutputSerialization.builder()
        .json(JSONOutput.builder().recordDelimiter(null).build())
        .build();

    // Build the SelectObjectContentRequest.
    SelectObjectContentRequest select = SelectObjectContentRequest.builder()
        .bucket(BUCKET_NAME)
        .key(FILE_JSON)
        .expression(queryExpression)
        .expressionType(ExpressionType.SQL)
        .inputSerialization(inputSerialization)
        .outputSerialization(outputSerialization)
        .build();

    EventStreamInfo eventStreamInfo = new EventStreamInfo();
    // Call the selectObjectContent method with the request and a response
handler.
```



```

    // Supply an EventStreamInfo object to the response handler to gather
    records and information from the response.
    s3AsyncClient.selectObjectContent(select,
    buildResponseHandler(eventStreamInfo)).join();

    // Log out information gathered while processing the response stream.
    long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record ->
        record.split("\n").length
    ).sum();
    logger.info("Total records {}: {}", fileType, recordCount);
    logger.info("Visitor onRecords for fileType {} called {} times", fileType,
    eventStreamInfo.getCountOnRecordsCalled());
    logger.info("Visitor onStats for fileType {}, {}", fileType,
    eventStreamInfo.getStats());
    logger.info("Visitor onContinuations for fileType {}, {}", fileType,
    eventStreamInfo.getCountContinuationEvents());
    return eventStreamInfo;
}

static SelectObjectContentResponseHandler buildResponseHandler(EventStreamInfo
eventStreamInfo) {
    // Use a Visitor to process the response stream. This visitor logs
    information and gathers details while processing.
    final SelectObjectContentResponseHandler.Visitor visitor =
    SelectObjectContentResponseHandler.Visitor.builder()
        .onRecords(r -> {
            logger.info("Record event received.");
            eventStreamInfo.addRecord(r.payload().asUtf8String());
            eventStreamInfo.incrementOnRecordsCalled();
        })
        .onCont(ce -> {
            logger.info("Continuation event received.");
            eventStreamInfo.incrementContinuationEvents();
        })
        .onProgress(pe -> {
            Progress progress = pe.details();
            logger.info("Progress event received:\n bytesScanned:
    {} \n bytesProcessed: {} \n bytesReturned: {}",
                progress.bytesScanned(),
                progress.bytesProcessed(),
                progress.bytesReturned());
        })
        .onEnd(ee -> logger.info("End event received."))
        .onStats(se -> {

```

```
        logger.info("Stats event received.");
        eventStreamInfo.addStats(se.details());
    })
    .build();

    // Build the SelectObjectContentResponseHandler with the visitor that
processes the stream.
    return SelectObjectContentResponseHandler.builder()
        .subscriber(visitor).build();
}

// The EventStreamInfo class is used to store information gathered while
processing the response stream.
static class EventStreamInfo {
    private final List<String> records = new ArrayList<>();
    private Integer countOnRecordsCalled = 0;
    private Integer countContinuationEvents = 0;
    private Stats stats;

    void incrementOnRecordsCalled() {
        countOnRecordsCalled++;
    }

    void incrementContinuationEvents() {
        countContinuationEvents++;
    }

    void addRecord(String record) {
        records.add(record);
    }

    void addStats(Stats stats) {
        this.stats = stats;
    }

    public List<String> getRecords() {
        return records;
    }

    public Integer getCountOnRecordsCalled() {
        return countOnRecordsCalled;
    }

    public Integer getCountContinuationEvents() {
```

```
        return countContinuationEvents;
    }

    public Stats getStats() {
        return stats;
    }
}
```

- Weitere API-Informationen finden Sie unter [SelectObjectContent](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Szenarien

### Eine vorsignierte URL erstellen

Das folgende Codebeispiel zeigt, wie Sie eine vorsignierte URL für Amazon S3 erstellen und ein Objekt hochladen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Generieren Sie eine vorsignierte URL für ein Objekt und laden Sie sie dann herunter (GET-Anforderung).

Importiert.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
```

```
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.UUID;
```

Generieren Sie die URL.

```
/* Create a pre-signed URL to download an object in a subsequent GET request. */
public String createPresignedGetUrl(String bucketName, String keyName) {
    try (S3Presigner presigner = S3Presigner.create()) {

        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        GetObjectPresignRequest presignRequest =
        GetObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // The URL will
            expire in 10 minutes.
            .getObjectRequest(objectRequest)
            .build();

        PresignedGetObjectRequest presignedRequest =
        presigner.presignGetObject(presignRequest);
        logger.info("Presigned URL: [{}]", presignedRequest.url().toString());
    }
}
```

```

        logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}

```

Laden Sie das Objekt mit einem der folgenden drei Ansätze herunter.

Verwenden Sie die Klasse JDK `URLConnection` (seit v1.1), um den Download durchzuführen.

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
public byte[] useHttpURLConnectionToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setRequestMethod("GET");
        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            IoUtils.copy(content, byteArrayOutputStream);
        }
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}

```

Verwenden Sie die Klasse JDK `HttpClient` (seit v11), um den Download durchzuführen.

```

/* Use the JDK HttpClient (since v11) class to do the download. */
public byte[] useHttpClientToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

```

```

HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
HttpClient httpClient = HttpClient.newHttpClient();
try {
    URL presignedUrl = new URL(presignedUrlString);
    HttpResponse<InputStream> response = httpClient.send(requestBuilder
        .uri(presignedUrl.toURI())
        .GET()
        .build(),
        HttpResponse.BodyHandlers.ofInputStream());

    IoUtils.copy(response.body(), byteArrayOutputStream);

    logger.info("HTTP response code is " + response.statusCode());
} catch (URISyntaxException | InterruptedException | IOException e) {
    logger.error(e.getMessage(), e);
}
return byteArrayOutputStream.toByteArray();
}

```

Verwenden Sie die `SdkHttpClient` Klasse AWS SDK for Java, um den Download durchzuführen.

```

/* Use the AWS SDK for Java SdkHttpClient class to do the download. */
public byte[] useSdkHttpClientToPut(String presignedUrlString) {

    ByteArrayOutputStream byteArrayOutputStream = new
    ByteArrayOutputStream(); // Capture the response body to a byte array.
    try {
        URL presignedUrl = new URL(presignedUrlString);
        SdkHttpRequest request = SdkHttpRequest.builder()
            .method(SdkHttpMethod.GET)
            .uri(presignedUrl.toURI())
            .build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
            sdkHttpClient.prepareRequest(executeRequest).call();

```

```

        response.responseBody().ifPresentOrElse(
            abortableInputStream -> {
                try {
                    IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            },
            () -> logger.error("No response body."));

        logger.info("HTTP Response code is {}",
response.httpResponse().statusCode());
    }
} catch (URISyntaxException | IOException e) {
    logger.error(e.getMessage(), e);
}
return byteArrayOutputStream.toByteArray();
}

```

Generieren Sie eine vorsignierte URL für einen Upload und laden Sie dann eine Datei hoch (PUT-Anforderung).

Importiert.

```

import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;

```

```
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

Generieren Sie die URL.

```
/* Create a presigned URL to use in a subsequent PUT request */
public String createPresignedUrl(String bucketName, String keyName, Map<String,
String> metadata) {
    try (S3Presigner presigner = S3Presigner.create()) {

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .metadata(metadata)
            .build();

        PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // The URL expires
in 10 minutes.
            .putObjectRequest(objectRequest)
            .build();

        PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
        String myURL = presignedRequest.url().toString();
        logger.info("Presigned URL to upload a file to: [{}]", myURL);
```



```

        logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}

```

Laden Sie ein Dateiojekt hoch, indem Sie einen der folgenden drei Ansätze verwenden.

Verwenden Sie die Klasse JDK `URLConnection` (seit v1.1), um den Upload durchzuführen.

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
public void useURLConnectionToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setDoOutput(true);
        metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-meta-" +
k, v));

        connection.setRequestMethod("PUT");
        OutputStream out = connection.getOutputStream();

        try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
            FileChannel inChannel = file.getChannel()) {
            ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is 8k

            while (inChannel.read(buffer) > 0) {
                buffer.flip();
                for (int i = 0; i < buffer.limit(); i++) {
                    out.write(buffer.get());
                }
                buffer.clear();
            }
        } catch (IOException e) {
            logger.error(e.getMessage(), e);
        }

        out.close();
        connection.getResponseCode();
        logger.info("HTTP response code is " + connection.getResponseCode());
    }
}

```

```

    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
}

```

Verwenden Sie die Klasse `JDK HttpClient` (seit v11), um den Upload durchzuführen.

```

/* Use the JDK HttpClient (since v11) class to do the upload. */
public void useHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        final HttpResponse<Void> response = httpClient.send(requestBuilder
            .uri(new URL(presignedUrlString).toURI())
            .PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI())))
            .build(),
            HttpResponse.BodyHandlers.discarding());

        logger.info("HTTP response code is " + response.statusCode());

    } catch (URISyntaxException | InterruptedException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}

```

Verwenden Sie die `SdkHttpClient` Klasse AWS für Java V2, um den Upload durchzuführen.

```

/* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    try {
        URL presignedUrl = new URL(presignedUrlString);

```

```
        SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
            .method(SdkHttpMethod.PUT)
            .uri(presignedUrl.toURI());
        // Add headers
        metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" + k,
v));

        // Finish building the request.
        SdkHttpRequest request = requestBuilder.build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
            logger.info("Response code: {}",
response.httpResponse().statusCode());
        }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}
```

## Erste Schritte mit Buckets und Objekten

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 Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an Amazon S3 bucket.
 * 2. Uploads an object to the bucket.
 * 3. Downloads the object to another local file.
 * 4. Uploads an object using multipart upload.
 * 5. List all objects located in the Amazon S3 bucket.
 * 6. Copies the object to another Amazon S3 bucket.
 * 7. Deletes the object from the Amazon S3 bucket.
 * 8. Deletes the Amazon S3 bucket.
 */

public class S3Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException {
        final String usage = ""

            Usage:
                <bucketName> <key> <objectPath> <savePath> <toBucket>

            Where:
                bucketName - The Amazon S3 bucket to create.
                key - The key to use.
                objectPath - The path where the file is located (for example,
                C:/AWS/book2.pdf).
    }
}
```

```
        savePath - The path where the file is saved after it's
downloaded (for example, C:/AWS/book2.pdf).
        toBucket - An Amazon S3 bucket to where an object is copied to
(for example, C:/AWS/book2.pdf).\s
        """;

    if (args.length != 5) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String key = args[1];
    String objectPath = args[2];
    String savePath = args[3];
    String toBucket = args[4];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon S3 example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Create an Amazon S3 bucket.");
    createBucket(s3, bucketName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Update a local file to the Amazon S3 bucket.");
    uploadLocalFile(s3, bucketName, key, objectPath);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. Download the object to another local file.");
    getObjectBytes(s3, bucketName, key, savePath);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. Perform a multipart upload.");
    String multipartKey = "multiPartKey";
    multipartUpload(s3, toBucket, multipartKey);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. List all objects located in the Amazon S3 bucket.");
        listAllObjects(s3, bucketName);
        anotherListExample(s3, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Copy the object to another Amazon S3 bucket.");
        copyBucketObject(s3, bucketName, key, toBucket);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the object from the Amazon S3 bucket.");
        deleteObjectFromBucket(s3, bucketName, key);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Delete the Amazon S3 bucket.");
        deleteBucket(s3, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon S3 operations were successfully performed");
        System.out.println(DASHES);
        s3.close();
    }

    // Create a bucket by using a S3Waiter object.
    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                .bucket(bucketName)
                .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            // Wait until the bucket is created and print out the response.
```

```
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteBucket(S3Client client, String bucket) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucket)
        .build();

    client.deleteBucket(deleteBucketRequest);
    System.out.println(bucket + " was deleted.");
}

/**
 * Upload an object in parts.
 */
public static void multipartUpload(S3Client s3, String bucketName, String key) {
    int mB = 1024 * 1024;
    // First create a multipart upload and get the upload id.
    CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
    String uploadId = response.uploadId();
    System.out.println(uploadId);

    // Upload all the different parts of the object.
    UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
        .bucket(bucketName)
        .key(key)
        .uploadId(uploadId)
        .partNumber(1).build();
```

```
String etag1 = s3.uploadPart(uploadPartRequest1,
    RequestBody.fromByteBuffer(getRandomByteBuffer(5 * MB)))
    .eTag();
CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
    .uploadId(uploadId)
    .partNumber(2).build();
String etag2 = s3.uploadPart(uploadPartRequest2,
    RequestBody.fromByteBuffer(getRandomByteBuffer(3 * MB)))
    .eTag();
CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

// Call completeMultipartUpload operation to tell S3 to merge all uploaded
// parts and finish the multipart operation.
CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
    .parts(part1, part2)
    .build();

CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)
    .multipartUpload(completedMultipartUpload)
    .build();

s3.completeMultipartUpload(completeMultipartUploadRequest);
}

private static ByteBuffer getRandomByteBuffer(int size) {
    byte[] b = new byte[size];
    new Random().nextBytes(b);
    return ByteBuffer.wrap(b);
}

public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
    try {
        GetObjectRequest objectRequest = GetObjectRequest
```



```
        .builder()
        .key(keyName)
        .bucket(bucketName)
        .build();

    ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
    byte[] data = objectBytes.asByteArray();

    // Write the data to a local file.
    File myFile = new File(path);
    OutputStream os = new FileOutputStream(myFile);
    os.write(data);
    System.out.println("Successfully obtained bytes from an S3 object");
    os.close();

} catch (IOException ex) {
    ex.printStackTrace();
} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void uploadLocalFile(S3Client s3, String bucketName, String key,
String objectPath) {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    s3.putObject(objectRequest, RequestBody.fromFile(new File(objectPath)));
}

public static void listAllObjects(S3Client s3, String bucketName) {
    ListObjectsV2Request listObjectsReqManual = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .maxKeys(1)
        .build();

    boolean done = false;
    while (!done) {
        ListObjectsV2Response listObjResponse =
s3.listObjectsV2(listObjectsReqManual);
```

```
        for (S3Object content : listObjResponse.contents()) {
            System.out.println(content.key());
        }

        if (listObjResponse.nextContinuationToken() == null) {
            done = true;
        }

        listObjectsReqManual = listObjectsReqManual.toBuilder()
            .continuationToken(listObjResponse.nextContinuationToken())
            .build();
    }
}

public static void anotherListExample(S3Client s3, String bucketName) {
    ListObjectsV2Request listReq = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .maxKeys(1)
        .build();

    ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);

    // Process response pages.
    listRes.stream()
        .flatMap(r -> r.contents().stream())
        .forEach(content -> System.out.println(" Key: " + content.key() + "
size = " + content.size()));

    // Helper method to work with paginated collection of items directly.
    listRes.contents().stream()
        .forEach(content -> System.out.println(" Key: " + content.key() + "
size = " + content.size()));

    for (S3Object content : listRes.contents()) {
        System.out.println(" Key: " + content.key() + " size = " +
content.size());
    }
}

public static void deleteObjectFromBucket(S3Client s3, String bucketName, String
key) {
    DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
```

```
        .build();

        s3.deleteObject(deleteObjectRequest);
        System.out.println(key + " was deleted");
    }

    public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
        String encodedUrl = null;
        try {
            encodedUrl = URLEncoder.encode(fromBucket + "/" + objectKey,
StandardCharsets.UTF_8.toString());
        } catch (UnsupportedEncodingException e) {
            System.out.println("URL could not be encoded: " + e.getMessage());
        }
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .copySource(encodedUrl)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
            .build();

        try {
            CopyObjectResponse copyRes = s3.copyObject(copyReq);
            System.out.println("The " + objectKey + " was copied to " + toBucket);
            return copyRes.copyObjectResult().toString();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)

- [ListObjectsV2](#)
- [PutObject](#)

## URIs analysieren

Das folgende Codebeispiel zeigt, wie Sie Amazon S3-URIs analysieren, um wichtige Komponenten wie den Bucket-Namen und Objektschlüssel zu extrahieren.

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Analysieren Sie eine Amazon S3-URI mithilfe der [S3Uri-Klasse](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.S3Uri;
import software.amazon.awssdk.services.s3.S3Utilities;

import java.net.URI;
import java.util.List;
import java.util.Map;

/**
 *
 * @param s3Client - An S3Client through which you acquire an S3Uri instance.
 * @param s3objectUrl - A complex URL (String) that is used to demonstrate S3Uri
 * capabilities.
 */
public static void parseS3UriExample(S3Client s3Client, String s3objectUrl) {
    logger.info(s3objectUrl);
    // Console output:
    // 'https://s3.us-west-1.amazonaws.com/myBucket/resources/doc.txt?
versionId=abc123&partNumber=77&partNumber=88'.
```

```
// Create an S3Utilities object using the configuration of the s3Client.
S3Utilities s3Utilities = s3Client.utilities();

// From a String URL create a URI object to pass to the parseUri() method.
URI uri = URI.create(s3ObjectUrl);
S3Uri s3Uri = s3Utilities.parseUri(uri);

// If the URI contains no value for the Region, bucket or key, the SDK
returns
// an empty Optional.
// The SDK returns decoded URI values.

Region region = s3Uri.region().orElse(null);
log("region", region);
// Console output: 'region: us-west-1'.

String bucket = s3Uri.bucket().orElse(null);
log("bucket", bucket);
// Console output: 'bucket: myBucket'.

String key = s3Uri.key().orElse(null);
log("key", key);
// Console output: 'key: resources/doc.txt'.

Boolean isPathStyle = s3Uri.isPathStyle();
log("isPathStyle", isPathStyle);
// Console output: 'isPathStyle: true'.

// If the URI contains no query parameters, the SDK returns an empty map.
Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();
log("rawQueryParameters", queryParams);
// Console output: 'rawQueryParameters: {versionId=[abc123], partNumber=[77,
// 88]}'

// Retrieve the first or all values for a query parameter as shown in the
// following code.
String versionId =
s3Uri.firstMatchingRawQueryParameter("versionId").orElse(null);
log("firstMatchingRawQueryParameter-versionId", versionId);
// Console output: 'firstMatchingRawQueryParameter-versionId: abc123'.

String partNumber =
s3Uri.firstMatchingRawQueryParameter("partNumber").orElse(null);
log("firstMatchingRawQueryParameter-partNumber", partNumber);
```

```

// Console output: 'firstMatchingRawQueryParameter-partNumber: 77'.

List<String> partNumbers =
s3Uri.firstMatchingRawQueryParameters("partNumber");
log("firstMatchingRawQueryParameter", partNumbers);
// Console output: 'firstMatchingRawQueryParameter: [77, 88]'.

/*
 * Object keys and query parameters with reserved or unsafe characters, must
be
 * URL-encoded.
 * For example replace whitespace " " with "%20".
 * Valid:
 * "https://s3.us-west-1.amazonaws.com/myBucket/object%20key?query=
%5Bbrackets%5D"
 * Invalid:
 * "https://s3.us-west-1.amazonaws.com/myBucket/object key?query=[brackets]"
 *
 * Virtual-hosted-style URIs with bucket names that contain a dot, ".", the
dot
 * must not be URL-encoded.
 * Valid: "https://my.Bucket.s3.us-west-1.amazonaws.com/key"
 * Invalid: "https://my%2EBucket.s3.us-west-1.amazonaws.com/key"
 */
}

private static void log(String s3UriElement, Object element) {
    if (element == null) {
        logger.info("{}: {}", s3UriElement, "null");
    } else {
        logger.info("{}: {}", s3UriElement, element.toString());
    }
}
}

```

## Durchführen eines mehrteiligen Uploads

Das folgende Codebeispiel zeigt, wie Sie einen mehrteiligen Upload in ein Amazon-S3-Objekt durchführen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

In den Codebeispielen werden folgende Importe verwendet.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
```

Verwenden Sie den [S3-Transfer-Manager](#) zusätzlich zum [AWS -CRT-basierten S3-Client](#), um einen mehrteiligen Upload auf transparente Weise durchzuführen, wenn die Größe des Inhalts einen Schwellenwert überschreitet. Der Standardschwellenwert beträgt 8 MB.

```
public void multipartUploadWithTransferManager(String filePath) {
```

```

S3TransferManager transferManager = S3TransferManager.create();
UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
    .putObjectRequest(b -> b
        .bucket(bucketName)
        .key(key))
    .source(Paths.get(filePath))
    .build();
FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
fileUpload.completionFuture().join();
transferManager.close();
}

```

Verwenden Sie die [S3Client-API](#) oder (S3AsyncClient API), um einen mehrteiligen Upload durchzuführen.

```

public void multipartUploadWithS3Client(String filePath) {

    // Initiate the multipart upload.
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
    .bucket(bucketName)
    .key(key));
    String uploadId = createMultipartUploadResponse.uploadId();

    // Upload the parts of the file.
    int partNumber = 1;
    List<CompletedPart> completedParts = new ArrayList<>();
    ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

    try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
        long fileSize = file.length();
        int position = 0;
        while (position < fileSize) {
            file.seek(position);
            int read = file.getChannel().read(bb);

            bb.flip(); // Swap position and limit before reading from the
buffer.

            UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)

```



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

    UploadPartResponse partResponse = s3Client.uploadPart(
        uploadPartRequest,
        RequestBody.fromByteBuffer(bb));

    CompletedPart part = CompletedPart.builder()
        .partNumber(partNumber)
        .eTag(partResponse.eTag())
        .build();
    completedParts.add(part);

    bb.clear();
    position += read;
    partNumber++;
}
} catch (IOException e) {
    logger.error(e.getMessage());
}

// Complete the multipart upload.
s3Client.completeMultipartUpload(b -> b
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [CompleteMultipartUpload](#)
  - [CreateMultipartUpload](#)
  - [UploadPart](#)

## Hoch- oder Herunterladen großer Dateien

Das folgende Codebeispiel zeigt, wie Sie große Dateien zu und von Amazon S3 hoch- oder herunterladen.

Weitere Informationen finden Sie unter [Hochladen eines Objekts mit Multipart-Upload](#).

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Rufen Sie Funktionen auf, die Dateien mithilfe von S3 in und aus einem S3S3TransferManager-Bucket übertragen.

```
public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
    URI destinationPathURI, String bucketName) {
    DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
    .destination(Paths.get(destinationPathURI))
    .bucket(bucketName)
    .build());
    CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryDownload.failedTransfers().size();
}
```

Laden Sie ein ganzes lokales Verzeichnis hoch.

```
public Integer uploadDirectory(S3TransferManager transferManager,
    URI sourceDirectory, String bucketName) {
    DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
    .source(Paths.get(sourceDirectory))
    .bucket(bucketName)
    .build());

    CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
    completedDirectoryUpload.failedTransfers()
```

```
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryUpload.failedTransfers().size();
}
```

Laden Sie eine einzelne Datei hoch.

```
public String uploadFile(S3TransferManager transferManager, String bucketName,
    String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create())
        .source(Paths.get(filePathURI))
        .build();

    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

    CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
    return uploadResult.response().eTag();
}
```

## Stream unbekannter Größe hochladen

Im folgenden Codebeispiel wird veranschaulicht, wie Sie einen Stream unbekannter Größe in ein Amazon S3-Objekt hochladen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Verwenden Sie den [AWS -CRT-basierten S3-Client](#).

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

/**
 * @param s3CrtAsyncClient - To upload content from a stream of unknown size,
 * use the AWS CRT-based S3 client. For more information, see
 * https://docs.aws.amazon.com/sdk-for-java/latest/
 \* developer-guide/crt-based-s3-client.html.
 * @param bucketName - The name of the bucket.
 * @param key - The name of the object.
 * @return software.amazon.awssdk.services.s3.model.PutObjectResponse - Returns
 * metadata pertaining to the put object operation.
 */
public PutObjectResponse putObjectFromStream(S3AsyncClient s3CrtAsyncClient,
String bucketName, String key) {

    BlockingInputStreamAsyncRequestBody body =
        AsyncRequestBody.forBlockingInputStream(null); // 'null' indicates a
    stream will be provided later.

    CompletableFuture<PutObjectResponse> responseFuture =
        s3CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
    body);

    // AsyncExampleUtils.randomString() returns a random string up to 100
    characters.
    String randomString = AsyncExampleUtils.randomString();
    logger.info("random string to upload: {}: length={}", randomString,
    randomString.length());

    // Provide the stream of data to be uploaded.
    body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

    PutObjectResponse response = responseFuture.join(); // Wait for the
    response.
    logger.info("Object {} uploaded to bucket {}.", key, bucketName);
    return response;
}
```

```
}  
}
```

Verwenden Sie den [Amazon-S3-Transfer-Manager](#).

```
import com.example.s3.util.AsyncExampleUtils;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import software.amazon.awssdk.core.async.AsyncRequestBody;  
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;  
import software.amazon.awssdk.core.exception.SdkException;  
import software.amazon.awssdk.transfer.s3.S3TransferManager;  
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;  
import software.amazon.awssdk.transfer.s3.model.Upload;  
  
import java.io.ByteArrayInputStream;  
import java.util.UUID;  
  
/**  
 * @param transferManager - To upload content from a stream of unknown size, use  
 the S3TransferManager based on the AWS CRT-based S3 client.  
 * For more information, see https://docs.aws.amazon.com/  
sdk-for-java/latest/developer-guide/transfer-manager.html.  
 * @param bucketName - The name of the bucket.  
 * @param key - The name of the object.  
 * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The  
 result of the completed upload.  
 */  
public CompletedUpload uploadStream(S3TransferManager transferManager, String  
bucketName, String key) {  
  
    BlockingInputStreamAsyncRequestBody body =  
        AsyncRequestBody.forBlockingInputStream(null); // 'null' indicates a  
stream will be provided later.  
  
    Upload upload = transferManager.upload(builder -> builder  
        .requestBody(body)  
        .putObjectRequest(req -> req.bucket(bucketName).key(key))  
        .build());  
  
    // AsyncExampleUtils.randomString() returns a random string up to 100  
characters.
```

```
String randomString = AsyncExampleUtils.randomString();
logger.info("random string to upload: {}: length={}", randomString,
randomString.length());

// Provide the stream of data to be uploaded.
body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

return upload.completionFuture().join();
}
}
```

## Verwenden der Prüfsummen

Das folgende Codebeispiel zeigt, wie Sie Prüfsummen verwenden, um mit einem Amazon-S3-Objekt zu arbeiten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

In den Codebeispielen wird eine Teilmenge der folgenden Importe verwendet.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.ChecksumMode;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
```

```
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.security.DigestInputStream;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
```

Geben Sie einen Prüfsummenalgorithmus für die `putObject`-Methode an, wenn Sie [PutObjectRequest erstellen](#).

```
public void putObjectWithChecksum() {
    s3Client.putObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(ChecksumAlgorithm.CRC32),
        RequestBody.fromString("This is a test"));
}
```

Überprüfen Sie die Prüfsumme für die `getObject` Methode, wenn Sie [die erstellen](#) [GetObjectRequest](#).

```
public GetObjectResponse getObjectWithChecksum() {
    return s3Client.getObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumMode(ChecksumMode.ENABLED))
```

```
        .response();  
    }
```

Berechnen Sie im Voraus eine Prüfsumme für die `putObject`-Methode, wenn Sie [PutObjectRequest erstellen](#).

```
public void putObjectWithPrecalculatedChecksum(String filePath) {  
    String checksum = calculateChecksum(filePath, "SHA-256");  
  
    s3Client.putObject((b -> b  
        .bucket(bucketName)  
        .key(key)  
        .checksumSHA256(checksum)),  
        RequestBody.fromFile(Paths.get(filePath)));  
}
```

Verwenden Sie den [S3-Transfer-Manager](#) zusätzlich zum [AWS -CRT-basierten S3-Client](#), um einen mehrteiligen Upload auf transparente Weise durchzuführen, wenn die Größe des Inhalts einen Schwellenwert überschreitet. Der Standardschwellenwert beträgt 8 MB.

Sie können einen Prüfsummenalgorithmus angeben, den das SDK verwenden soll. Standardmäßig verwendet das SDK den CRC32-Algorithmus.

```
public void multipartUploadWithChecksumTm(String filePath) {  
    S3TransferManager transferManager = S3TransferManager.create();  
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()  
        .putObjectRequest(b -> b  
            .bucket(bucketName)  
            .key(key)  
            .checksumAlgorithm(ChecksumAlgorithm.SHA1))  
        .source(Paths.get(filePath))  
        .build();  
    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);  
    fileUpload.completionFuture().join();  
    transferManager.close();  
}
```

Verwenden Sie die [S3Client-API](#) oder (S3AsyncClient API), um einen mehrteiligen Upload durchzuführen. Wenn Sie eine zusätzliche Prüfsumme angeben, müssen Sie den Algorithmus



angeben, der bei der Initiierung des Uploads verwendet werden soll. Sie müssen auch den Algorithmus für jede Teilanforderung angeben und die für jedes Teil nach dem Hochladen berechnete Prüfsumme bereitstellen.

```
public void multipartUploadWithChecksumS3Client(String filePath) {
    ChecksumAlgorithm algorithm = ChecksumAlgorithm.CRC32;

    // Initiate the multipart upload.
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(algorithm)); // Checksum specified on initiation.
    String uploadId = createMultipartUploadResponse.uploadId();

    // Upload the parts of the file.
    int partNumber = 1;
    List<CompletedPart> completedParts = new ArrayList<>();
    ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

    try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
        long fileSize = file.length();
        int position = 0;
        while (position < fileSize) {
            file.seek(position);
            int read = file.getChannel().read(bb);

            bb.flip(); // Swap position and limit before reading from the
buffer.

            UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .checksumAlgorithm(algorithm) // Checksum specified on each
part.

                .partNumber(partNumber)
                .build();

            UploadPartResponse partResponse = s3Client.uploadPart(
                uploadPartRequest,
                RequestBody.fromByteBuffer(bb));

            CompletedPart part = CompletedPart.builder()
```

```

        .partNumber(partNumber)
        .checksumCRC32(partResponse.checksumCRC32()) // Provide the
calculated checksum.
        .eTag(partResponse.eTag())
        .build();
    completedParts.add(part);

    bb.clear();
    position += read;
    partNumber++;
}
} catch (IOException e) {
    System.err.println(e.getMessage());
}

// Complete the multipart upload.
s3Client.completeMultipartUpload(b -> b
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
}

```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [CompleteMultipartUpload](#)
  - [CreateMultipartUpload](#)
  - [UploadPart](#)

## Serverless-Beispiele

### Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

## Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Java

```
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
    com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotificat

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
            S3EventNotificationRecord record = s3event.getRecords().get(0);
            String srcBucket = record.getS3().getBucket().getName();
            String srcKey = record.getS3().getObject().getUrlDecodedKey();

            S3Client s3Client = S3Client.builder().build();
            HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
srcKey);

            logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
type " + headObject.contentType());

            return "Ok";
        } catch (Exception e) {
```

```
        throw new RuntimeException(e);
    }
}

private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
String key) {
    HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
        .bucket(bucket)
        .key(key)
        .build();
    return s3Client.headObject(headObjectRequest);
}
}
```

## S3-Glacier-Beispiele mit SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit S3 Glacier verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)

### Aktionen

#### Erstellen eines Tresors

Das folgende Codebeispiel zeigt, wie Sie einen Amazon S3-Glacier-Tresor erstellen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.CreateVaultRequest;
import software.amazon.awssdk.services.glacier.model.CreateVaultResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateVault {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <vaultName>

            Where:
                vaultName - The name of the vault to create.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
            .region(Region.US_EAST_1)
            .build();
```

```
        createGlacierVault(glacier, vaultName);
        glacier.close();
    }

    public static void createGlacierVault(GlacierClient glacier, String vaultName) {
        try {
            CreateVaultRequest vaultRequest = CreateVaultRequest.builder()
                .vaultName(vaultName)
                .build();

            CreateVaultResponse createVaultResult =
glacier.createVault(vaultRequest);
            System.out.println("The URI of the new vault is " +
createVaultResult.location());

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [CreateVault](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen eines Tresors

Das folgende Codebeispiel zeigt, wie Sie einen Amazon S3-Glacier-Tresor löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteVaultRequest;
```

```
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteVault {
    public static void main(String[] args) {

        final String usage = ""

            Usage:    <vaultName>

            Where:
                vaultName - The name of the vault to delete.\s
            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
            .region(Region.US_EAST_1)
            .build();

        deleteGlacierVault(glacier, vaultName);
        glacier.close();
    }

    public static void deleteGlacierVault(GlacierClient glacier, String vaultName) {
        try {
            DeleteVaultRequest delVaultRequest = DeleteVaultRequest.builder()
                .vaultName(vaultName)
                .build();

            glacier.deleteVault(delVaultRequest);
            System.out.println("The vault was deleted!");
        }
    }
}
```

```

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- Weitere API-Informationen finden Sie unter [DeleteVault](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen eines Archivs

Das folgende Codebeispiel zeigt, wie Sie ein Amazon S3-Glacier-Archiv löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteArchiveRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteArchive {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <vaultName> <accountId> <archiveId>

```



```
        Where:
            vaultName - The name of the vault that contains the archive to
delete.

            accountId - The account ID value.
            archiveId - The archive ID value.
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String vaultName = args[0];
    String accountId = args[1];
    String archiveId = args[2];
    GlacierClient glacier = GlacierClient.builder()
        .region(Region.US_EAST_1)
        .build();

    deleteGlacierArchive(glacier, vaultName, accountId, archiveId);
    glacier.close();
}

public static void deleteGlacierArchive(GlacierClient glacier, String vaultName,
String accountId,
    String archiveId) {
    try {
        DeleteArchiveRequest delArcRequest = DeleteArchiveRequest.builder()
            .vaultName(vaultName)
            .accountId(accountId)
            .archiveId(archiveId)
            .build();

        glacier.deleteArchive(delArcRequest);
        System.out.println("The archive was deleted.");

    } catch (GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [DeleteArchive](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Tresoren

Das folgende Codebeispiel zeigt, wie Sie Amazon S3-Glacier-Tresore auflisten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.model.ListVaultsRequest;
import software.amazon.awssdk.services.glacier.model.ListVaultsResponse;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DescribeVaultOutput;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListVaults {
    public static void main(String[] args) {
        GlacierClient glacier = GlacierClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllVault(glacier);
        glacier.close();
    }

    public static void listAllVault(GlacierClient glacier) {
```

```
boolean listComplete = false;
String newMarker = null;
int totalVaults = 0;
System.out.println("Your Amazon Glacier vaults:");
try {
    while (!listComplete) {
        ListVaultsResponse response = null;
        if (newMarker != null) {
            ListVaultsRequest request = ListVaultsRequest.builder()
                .marker(newMarker)
                .build();

            response = glacier.listVaults(request);
        } else {
            ListVaultsRequest request = ListVaultsRequest.builder()
                .build();
            response = glacier.listVaults(request);
        }

        List<DescribeVaultOutput> vaultList = response.vaultList();
        for (DescribeVaultOutput v : vaultList) {
            totalVaults += 1;
            System.out.println("* " + v.vaultName());
        }

        // Check for further results.
        newMarker = response.marker();
        if (newMarker == null) {
            listComplete = true;
        }
    }

    if (totalVaults == 0) {
        System.out.println("No vaults found.");
    }

} catch (GlacierException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [ListVaults](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Abrufen einer Tresorinventur

Das folgende Codebeispiel zeigt, wie Sie ein Amazon S3-Glacier-Tresorinventar abrufen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.JobParameters;
import software.amazon.awssdk.services.glacier.model.InitiateJobResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import software.amazon.awssdk.services.glacier.model.InitiateJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobResponse;
import software.amazon.awssdk.services.glacier.model.GetJobOutputRequest;
import software.amazon.awssdk.services.glacier.model.GetJobOutputResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ArchiveDownload {
    public static void main(String[] args) {
```

```
final String usage = ""

    Usage:    <vaultName> <accountId> <path>

    Where:
        vaultName - The name of the vault.
        accountId - The account ID value.
        path - The path where the file is written to.
    """;

if (args.length != 3) {
    System.out.println(usage);
    System.exit(1);
}

String vaultName = args[0];
String accountId = args[1];
String path = args[2];
GlacierClient glacier = GlacierClient.builder()
    .region(Region.US_EAST_1)
    .build();

String jobNum = createJob(glacier, vaultName, accountId);
checkJob(glacier, jobNum, vaultName, accountId, path);
glacier.close();
}

public static String createJob(GlacierClient glacier, String vaultName, String
accountId) {
    try {
        JobParameters job = JobParameters.builder()
            .type("inventory-retrieval")
            .build();

        InitiateJobRequest initJob = InitiateJobRequest.builder()
            .jobParameters(job)
            .accountId(accountId)
            .vaultName(vaultName)
            .build();

        InitiateJobResponse response = glacier.initiateJob(initJob);
        System.out.println("The job ID is: " + response.jobId());
        System.out.println("The relative URI path of the job is: " +
response.location());
    }
}
```

```
        return response.jobId();

    } catch (GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}

// Poll S3 Glacier = Polling a Job may take 4-6 hours according to the
// Documentation.
public static void checkJob(GlacierClient glacier, String jobId, String name,
String account, String path) {
    try {
        boolean finished = false;
        String jobStatus;
        int yy = 0;

        while (!finished) {
            DescribeJobRequest jobRequest = DescribeJobRequest.builder()
                .jobId(jobId)
                .accountId(account)
                .vaultName(name)
                .build();

            DescribeJobResponse response = glacier.describeJob(jobRequest);
            jobStatus = response.statusCodeAsString();

            if (jobStatus.compareTo("Succeeded") == 0)
                finished = true;
            else {
                System.out.println(yy + " status is: " + jobStatus);
                Thread.sleep(1000);
            }
            yy++;
        }

        System.out.println("Job has Succeeded");
        GetJobOutputRequest jobOutputRequest = GetJobOutputRequest.builder()
            .jobId(jobId)
            .vaultName(name)
            .accountId(account)
            .build();
```

```
        ResponseBytes<GetJobOutputResponse> objectBytes =
glacier.getJobOutputAsBytes(jobOutputRequest);
        // Write the data to a local file.
        byte[] data = objectBytes.asByteArray();
        File myFile = new File(path);
        OutputStream os = new FileOutputStream(myFile);
        os.write(data);
        System.out.println("Successfully obtained bytes from a Glacier vault");
        os.close();

    } catch (GlacierException | InterruptedException | IOException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [InitiateJob](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Hochladen eines Archivs in einen Tresor

Das folgende Codebeispiel zeigt, wie Sie ein Archiv in einen Amazon S3-Glacier-Tresor hochladen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.UploadArchiveRequest;
import software.amazon.awssdk.services.glacier.model.UploadArchiveResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.io.File;
import java.nio.file.Path;
```

```
import java.nio.file.Paths;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UploadArchive {

    static final int ONE_MB = 1024 * 1024;

    public static void main(String[] args) {
        final String usage = ""

            Usage:  <strPath> <vaultName>\s

            Where:
                strPath - The path to the archive to upload (for example, C:\\AWS
\\test.pdf).
                vaultName - The name of the vault.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String strPath = args[0];
        String vaultName = args[1];
        File myFile = new File(strPath);
        Path path = Paths.get(strPath);
        GlacierClient glacier = GlacierClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String archiveId = uploadContent(glacier, path, vaultName, myFile);
        System.out.println("The ID of the archived item is " + archiveId);
    }
}
```



```
        glacier.close();
    }

    public static String uploadContent(GlacierClient glacier, Path path, String
vaultName, File myFile) {
    // Get an SHA-256 tree hash value.
    String checkVal = computeSHA256(myFile);
    try {
        UploadArchiveRequest uploadRequest = UploadArchiveRequest.builder()
            .vaultName(vaultName)
            .checksum(checkVal)
            .build();

        UploadArchiveResponse res = glacier.uploadArchive(uploadRequest, path);
        return res.archiveId();

    } catch (GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

private static String computeSHA256(File inputFile) {
    try {
        byte[] treeHash = computeSHA256TreeHash(inputFile);
        System.out.printf("SHA-256 tree hash = %s\n", toHex(treeHash));
        return toHex(treeHash);

    } catch (IOException ioe) {
        System.err.format("Exception when reading from file %s: %s", inputFile,
ioe.getMessage());
        System.exit(-1);

    } catch (NoSuchAlgorithmException nsae) {
        System.err.format("Cannot locate MessageDigest algorithm for SHA-256:
%s", nsae.getMessage());
        System.exit(-1);
    }
    return "";
}

public static byte[] computeSHA256TreeHash(File inputFile) throws IOException,
NoSuchAlgorithmException {
```

```
        byte[][] chunkSHA256Hashes = getChunkSHA256Hashes(inputFile);
        return computeSHA256TreeHash(chunkSHA256Hashes);
    }

    /**
     * Computes an SHA256 checksum for each 1 MB chunk of the input file. This
     * includes the checksum for the last chunk, even if it's smaller than 1 MB.
     */
    public static byte[][] getChunkSHA256Hashes(File file) throws IOException,
        NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        long numChunks = file.length() / ONE_MB;
        if (file.length() % ONE_MB > 0) {
            numChunks++;
        }

        if (numChunks == 0) {
            return new byte[][] { md.digest() };
        }

        byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
        FileInputStream fileStream = null;

        try {
            fileStream = new FileInputStream(file);
            byte[] buff = new byte[ONE_MB];

            int bytesRead;
            int idx = 0;

            while ((bytesRead = fileStream.read(buff, 0, ONE_MB)) > 0) {
                md.reset();
                md.update(buff, 0, bytesRead);
                chunkSHA256Hashes[idx++] = md.digest();
            }

            return chunkSHA256Hashes;
        } finally {
            if (fileStream != null) {
                try {
                    fileStream.close();
                }
            }
        }
    }
}
```

```
        } catch (IOException ioe) {
            System.err.printf("Exception while closing %s.\n %s",
file.getName(),
                ioe.getMessage());
        }
    }
}

/**
 * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
 * checksums.
 */
public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
    throws NoSuchAlgorithmException {

    MessageDigest md = MessageDigest.getInstance("SHA-256");
    byte[][] prevLvlHashes = chunkSHA256Hashes;
    while (prevLvlHashes.length > 1) {
        int len = prevLvlHashes.length / 2;
        if (prevLvlHashes.length % 2 != 0) {
            len++;
        }

        byte[][] currLvlHashes = new byte[len][];
        int j = 0;
        for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {

            // If there are at least two elements remaining.
            if (prevLvlHashes.length - i > 1) {

                // Calculate a digest of the concatenated nodes.
                md.reset();
                md.update(prevLvlHashes[i]);
                md.update(prevLvlHashes[i + 1]);
                currLvlHashes[j] = md.digest();

            } else { // Take care of the remaining odd chunk
                currLvlHashes[j] = prevLvlHashes[i];
            }
        }

        prevLvlHashes = currLvlHashes;
    }
}
```

```
        return prevLvlHashes[0];
    }

    /**
     * Returns the hexadecimal representation of the input byte array
     */
    public static String toHex(byte[] data) {
        StringBuilder sb = new StringBuilder(data.length * 2);
        for (byte datum : data) {
            String hex = Integer.toHexString(datum & 0xFF);

            if (hex.length() == 1) {
                // Append leading zero.
                sb.append("0");
            }
            sb.append(hex);
        }
        return sb.toString().toLowerCase();
    }
}
```

- Weitere API-Informationen finden Sie unter [UploadArchive](#) in der APIAWS SDK for Java 2.x - Referenz für .

## SageMaker -Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit verwenden SageMaker.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Erste Schritte

## Hallo SageMaker

Die folgenden Codebeispiele veranschaulichen, wie Sie mit der Verwendung von SageMaker beginnen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloSageMaker {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SageMakerClient sageMakerClient = SageMakerClient.builder()
            .region(region)
            .build();

        listBooks(sageMakerClient);
        sageMakerClient.close();
    }

    public static void listBooks(SageMakerClient sageMakerClient) {
        try {
            ListNotebookInstancesResponse notebookInstancesResponse =
sageMakerClient.listNotebookInstances();
            List<NotebookInstanceSummary> items =
notebookInstancesResponse.notebookInstances();
            for (NotebookInstanceSummary item : items) {
                System.out.println("The notebook name is: " +
item.notebookInstanceName());
            }
        }
    }
}
```

```
        } catch (SageMakerException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [ListNotebookInstances](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Themen

- [Aktionen](#)
- [Szenarien](#)

## Aktionen

Erstellen Sie eine Pipeline

Das folgende Codebeispiel zeigt, wie Sie eine Pipeline in erstellen oder aktualisieren SageMaker.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
// Create a pipeline from the example pipeline JSON.
public static void setupPipeline(SageMakerClient sageMakerClient, String
filePath, String roleArn,
    String functionArn, String pipelineName) {
    System.out.println("Setting up the pipeline.");
    JSONParser parser = new JSONParser();

    // Read JSON and get pipeline definition.
    try (FileReader reader = new FileReader(filePath)) {
```

```
Object obj = parser.parse(reader);
JSONObject jsonObject = (JSONObject) obj;
JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
for (Object stepObj : stepsArray) {
    JSONObject step = (JSONObject) stepObj;
    if (step.containsKey("FunctionArn")) {
        step.put("FunctionArn", functionArn);
    }
}
System.out.println(jsonObject);

// Create the pipeline.
CreatePipelineRequest pipelineRequest = CreatePipelineRequest.builder()
    .pipelineDescription("Java SDK example pipeline")
    .roleArn(roleArn)
    .pipelineName(pipelineName)
    .pipelineDefinition(jsonObject.toString())
    .build();

sageMakerClient.createPipeline(pipelineRequest);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
} catch (IOException | ParseException e) {
    throw new RuntimeException(e);
}
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [CreatePipeline](#)
  - [UpdatePipeline](#)

## Pipeline löschen

Das folgende Codebeispiel zeigt, wie Sie eine Pipeline in löschen SageMaker.

## SDK für Java 2.x

**Note**

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
// Delete a SageMaker pipeline by name.
public static void deletePipeline(SageMakerClient sageMakerClient, String
pipelineName) {
    DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
        .pipelineName(pipelineName)
        .build();

    sageMakerClient.deletePipeline(pipelineRequest);
    System.out.println("*** Successfully deleted " + pipelineName);
}
```

- Weitere API-Informationen finden Sie unter [DeletePipeline](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beschreiben einer Pipeline-Ausführung

Das folgende Codebeispiel zeigt, wie Sie eine Pipeline-Ausführung in beschreiben SageMaker.

## SDK für Java 2.x

**Note**

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
// Check the status of a pipeline execution.
public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
String executionArn)
    throws InterruptedException {
    String status;
```



```
int index = 0;
do {
    DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
    .pipelineExecutionArn(executionArn)
    .build();

    DescribePipelineExecutionResponse response = sageMakerClient
    .describePipelineExecution(pipelineExecutionRequest);
    status = response.pipelineExecutionStatusAsString();
    System.out.println(index + ". The Status of the pipeline is " + status);
    TimeUnit.SECONDS.sleep(4);
    index++;
} while ("Executing".equals(status));
System.out.println("Pipeline finished with status " + status);
}
```

- Weitere API-Informationen finden Sie unter [DescribePipelineExecution](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Ausführen einer Pipeline

Das folgende Codebeispiel zeigt, wie Sie eine Pipeline-Ausführung in starten SageMaker.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
// Start a pipeline run with job configurations.
public static String executePipeline(SageMakerClient sageMakerClient, String
bucketName, String queueUrl,
    String roleArn, String pipelineName) {
    System.out.println("Starting pipeline execution.");
    String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
    String output = "s3://" + bucketName + "/outputfiles/";
    Gson gson = new GsonBuilder()
```

```
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting().create());

// Set up all parameters required to start the pipeline.
List<Parameter> parameters = new ArrayList<>();
Parameter para1 = Parameter.builder()
    .name("parameter_execution_role")
    .value(roleArn)
    .build();

Parameter para2 = Parameter.builder()
    .name("parameter_queue_url")
    .value(queueUrl)
    .build();

String inputJSON = "{\n" +
    "  \"DataSourceConfig\": {\n" +
    "    \"S3Data\": {\n" +
    "      \"S3Uri\": \"s3://" + bucketName + "/samplefiles/"
latlongtest.csv\"\n" +
    "    },\n" +
    "    \"Type\": \"S3_DATA\"\n" +
    "  },\n" +
    "  \"DocumentType\": \"CSV\"\n" +
    "}";

System.out.println(inputJSON);

Parameter para3 = Parameter.builder()
    .name("parameter_vej_input_config")
    .value(inputJSON)
    .build();

// Create an ExportVectorEnrichmentJobOutputConfig object.
VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
    .s3Uri(output)
    .build();

ExportVectorEnrichmentJobOutputConfig outputConfig =
ExportVectorEnrichmentJobOutputConfig.builder()
    .s3Data(jobS3Data)
    .build();

String gson4 = gson.toJson(outputConfig);
```

```
Parameter para4 = Parameter.builder()
    .name("parameter_vej_export_config")
    .value(gson4)
    .build();
System.out.println("parameter_vej_export_config:" +
gson.toJson(outputConfig));

// Create a VectorEnrichmentJobConfig object.
ReverseGeocodingConfig reverseGeocodingConfig =
ReverseGeocodingConfig.builder()
    .xAttributeName("Longitude")
    .yAttributeName("Latitude")
    .build();

VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
    .reverseGeocodingConfig(reverseGeocodingConfig)
    .build();

String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":
{\"XAttributeName\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}";
Parameter para5 = Parameter.builder()
    .name("parameter_step_1_vej_config")
    .value(para5JSON)
    .build();

System.out.println("parameter_step_1_vej_config:" + gson.toJson(jobConfig));
parameters.add(para1);
parameters.add(para2);
parameters.add(para3);
parameters.add(para4);
parameters.add(para5);

StartPipelineExecutionRequest pipelineExecutionRequest =
StartPipelineExecutionRequest.builder()
    .pipelineExecutionDescription("Created using Java SDK")
    .pipelineExecutionDisplayName(pipelineName + "-example-execution")
    .pipelineParameters(parameters)
    .pipelineName(pipelineName)
    .build();

StartPipelineExecutionResponse response =
sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
return response.pipelineExecutionArn();
}
```

- Weitere API-Informationen finden Sie unter [StartPipelineExecution](#) in der APIAWS SDK for Java 2.x -Referenz für .

## 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.
- Pipeline-Ausführung starten.
- Überwachen Sie den Status der Ausführung.
- Zeigen Sie die Ausgabe der Pipeline an.
- Bereinigen Sie Ressourcen.

Weitere Informationen finden Sie unter [Erstellen und Ausführen von SageMaker Pipelines mit - AWS SDKs auf Community.aws](#).

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public class SagemakerWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String eventSourceMapping = "";

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
            "Usage:\n" +
```

```

        "    <sageMakerRoleName> <lambdaRoleName> <functionFileLocation>
<functionName> <queueName> <bucketName> <lnglatData> <spatialPipelinePath>
<pipelineName>\n\n"
    +
    "Where:\n" +
    "    sageMakerRoleName - The name of the Amazon SageMaker role.\n\n"
+
    "    lambdaRoleName - The name of the AWS Lambda role.\n\n" +
    "    functionFileLocation - The file location where the JAR file
that represents the AWS Lambda function is located.\n\n"
    +
    "    functionName - The name of the AWS Lambda function (for
example, SageMakerExampleFunction).\n\n" +
    "    queueName - The name of the Amazon Simple Queue Service (Amazon
SQS) queue.\n\n" +
    "    bucketName - The name of the Amazon Simple Storage Service
(Amazon S3) bucket.\n\n" +
    "    lnglatData - The file location of the latlongtest.csv file
required for this use case.\n\n" +
    "    spatialPipelinePath - The file location of the
GeoSpatialPipeline.json file required for this use case.\n\n"
    +
    "    pipelineName - The name of the pipeline to create (for example,
sagemaker-sdk-example-pipeline).\n\n";

    if (args.length != 9) {
        System.out.println(usage);
        System.exit(1);
    }

    String sageMakerRoleName = args[0];
    String lambdaRoleName = args[1];
    String functionFileLocation = args[2];
    String functionName = args[3];
    String queueName = args[4];
    String bucketName = args[5];
    String lnglatData = args[6];
    String spatialPipelinePath = args[7];
    String pipelineName = args[8];
    String handlerName = "org.example.SageMakerLambdaFunction::handleRequest";

    Region region = Region.US_WEST_2;
    SageMakerClient sageMakerClient = SageMakerClient.builder()
        .region(region)

```

```
        .build();

    IAMClient iam = IAMClient.builder()
        .region(region)
        .build();

    LambdaClient lambdaClient = LambdaClient.builder()
        .region(region)
        .build();

    SQSClient sqsClient = SQSClient.builder()
        .region(region)
        .build();

    S3Client s3Client = S3Client.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon SageMaker pipeline example
scenario.");
    System.out.println(
        "\nThis example workflow will guide you through setting up and
running an " +
            "\nAmazon SageMaker pipeline. The pipeline uses an AWS
Lambda function and an " +
            "\nAmazon SQS Queue. It runs a vector enrichment reverse
geocode job to " +
            "\nreverse geocode addresses in an input file and store the
results in an export file.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("First, we will set up the roles, functions, and queue
needed by the SageMaker pipeline.");
    String lambdaRoleArn = checkLambdaRole(iam, lambdaRoleName);
    String sageMakerRoleArn = checkSageMakerRole(iam, sageMakerRoleName);

    String functionArn = checkFunction(lambdaClient, functionName,
functionFileLocation, lambdaRoleArn,
        handlerName);
    String queueUrl = checkQueue(sqsClient, lambdaClient, queueName,
functionName);
    System.out.println("The queue URL is " + queueUrl);
```

```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Setting up bucket " + bucketName);
if (!checkBucket(s3Client, bucketName)) {
    setupBucket(s3Client, bucketName);
    System.out.println("Put " + lnglatData + " into " + bucketName);
    putS3Object(s3Client, bucketName, "latlongtest.csv", lnglatData);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Now we can create and run our pipeline.");
setupPipeline(sageMakerClient, spatialPipelinePath, sageMakerRoleArn,
functionArn, pipelineName);
String pipelineExecutionARN = executePipeline(sageMakerClient, bucketName,
queueUrl, sageMakerRoleArn,
    pipelineName);
System.out.println("The pipeline execution ARN value is " +
pipelineExecutionARN);
waitForPipelineExecution(sageMakerClient, pipelineExecutionARN);
System.out.println("Getting output results " + bucketName);
getOutputResults(s3Client, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The pipeline has completed. To view the pipeline and
runs " +
    "in SageMaker Studio, follow these instructions:" +
    "\nhttps://docs.aws.amazon.com/sagemaker/latest/dg/pipelines-
studio.html");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Do you want to delete the AWS resources used in this
Workflow? (y/n)");
Scanner in = new Scanner(System.in);
String delResources = in.nextLine();
if (delResources.compareTo("y") == 0) {
    System.out.println("Lets clean up the AWS resources. Wait 30 seconds");
    TimeUnit.SECONDS.sleep(30);
    deleteEventSourceMapping(lambdaClient);
    deleteSQSQueue(sqsClient, queueName);
    listBucketObjects(s3Client, bucketName);
}
```

```
        deleteBucket(s3Client, bucketName);
        deleteLambdaFunction(lambdaClient, functionName);
        deleteLambdaRole(iam, lambdaRoleName);
        deleteSageMakerRole(iam, sageMakerRoleName);
        deletePipeline(sageMakerClient, pipelineName);
    } else {
        System.out.println("The AWS Resources were not deleted!");
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("SageMaker pipeline scenario is complete.");
    System.out.println(DASHES);
}

private static void readObject(S3Client s3Client, String bucketName, String key)
{
    System.out.println("Output file contents: \n");
    GetObjectRequest objectRequest = GetObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
    byte[] byteArray = objectBytes.asByteArray();
    String text = new String(byteArray, StandardCharsets.UTF_8);
    System.out.println("Text output: " + text);
}

// Display some results from the output directory.
public static void getOutputResults(S3Client s3Client, String bucketName) {
    System.out.println("Getting output results {"bucketName}.");
    ListObjectsRequest listObjectsRequest = ListObjectsRequest.builder()
        .bucket(bucketName)
        .prefix("outputfiles/")
        .build();

    ListObjectsResponse response = s3Client.listObjects(listObjectsRequest);
    List<S3Object> s3Objects = response.contents();
    for (S3Object object : s3Objects) {
        readObject(s3Client, bucketName, object.key());
    }
}
```



```
// Check the status of a pipeline execution.
public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
String executionArn)
    throws InterruptedException {
    String status;
    int index = 0;
    do {
        DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
            .pipelineExecutionArn(executionArn)
            .build();

        DescribePipelineExecutionResponse response = sageMakerClient
            .describePipelineExecution(pipelineExecutionRequest);
        status = response.pipelineExecutionStatusAsString();
        System.out.println(index + ". The Status of the pipeline is " + status);
        TimeUnit.SECONDS.sleep(4);
        index++;
    } while ("Executing".equals(status));
    System.out.println("Pipeline finished with status " + status);
}

// Delete a SageMaker pipeline by name.
public static void deletePipeline(SageMakerClient sageMakerClient, String
pipelineName) {
    DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
        .pipelineName(pipelineName)
        .build();

    sageMakerClient.deletePipeline(pipelineRequest);
    System.out.println("*** Successfully deleted " + pipelineName);
}

// Create a pipeline from the example pipeline JSON.
public static void setupPipeline(SageMakerClient sageMakerClient, String
filePath, String roleArn,
    String functionArn, String pipelineName) {
    System.out.println("Setting up the pipeline.");
    JSONParser parser = new JSONParser();

    // Read JSON and get pipeline definition.
    try (FileReader reader = new FileReader(filePath)) {
        Object obj = parser.parse(reader);
    }
}
```

```
JSONObject jsonObject = (JSONObject) obj;
JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
for (Object stepObj : stepsArray) {
    JSONObject step = (JSONObject) stepObj;
    if (step.containsKey("FunctionArn")) {
        step.put("FunctionArn", functionArn);
    }
}
System.out.println(jsonObject);

// Create the pipeline.
CreatePipelineRequest pipelineRequest = CreatePipelineRequest.builder()
    .pipelineDescription("Java SDK example pipeline")
    .roleArn(roleArn)
    .pipelineName(pipelineName)
    .pipelineDefinition(jsonObject.toString())
    .build();

sageMakerClient.createPipeline(pipelineRequest);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
} catch (IOException | ParseException e) {
    throw new RuntimeException(e);
}
}

// Start a pipeline run with job configurations.
public static String executePipeline(SageMakerClient sageMakerClient, String
bucketName, String queueUrl,
    String roleArn, String pipelineName) {
    System.out.println("Starting pipeline execution.");
    String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
    String output = "s3://" + bucketName + "/outputfiles/";
    Gson gson = new GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting().create();

    // Set up all parameters required to start the pipeline.
    List<Parameter> parameters = new ArrayList<>();
    Parameter para1 = Parameter.builder()
        .name("parameter_execution_role")
```

```
        .value(roleArn)
        .build();

    Parameter para2 = Parameter.builder()
        .name("parameter_queue_url")
        .value(queueUrl)
        .build();

    String inputJSON = "{\n" +
        "  \"DataSourceConfig\": {\n" +
        "    \"S3Data\": {\n" +
        "      \"S3Uri\": \"s3://\" + bucketName + \"/samplefiles/"
latlongtest.csv\"\n" +
        "    },\n" +
        "    \"Type\": \"S3_DATA\"\n" +
        "  },\n" +
        "  \"DocumentType\": \"CSV\"\n" +
        "}";

    System.out.println(inputJSON);

    Parameter para3 = Parameter.builder()
        .name("parameter_vej_input_config")
        .value(inputJSON)
        .build();

    // Create an ExportVectorEnrichmentJobOutputConfig object.
    VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
        .s3Uri(output)
        .build();

    ExportVectorEnrichmentJobOutputConfig outputConfig =
ExportVectorEnrichmentJobOutputConfig.builder()
        .s3Data(jobS3Data)
        .build();

    String gson4 = gson.toJson(outputConfig);
    Parameter para4 = Parameter.builder()
        .name("parameter_vej_export_config")
        .value(gson4)
        .build();

    System.out.println("parameter_vej_export_config:" +
gson.toJson(outputConfig));
```

```
// Create a VectorEnrichmentJobConfig object.
ReverseGeocodingConfig reverseGeocodingConfig =
ReverseGeocodingConfig.builder()
    .xAttributeName("Longitude")
    .yAttributeName("Latitude")
    .build();

VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
    .reverseGeocodingConfig(reverseGeocodingConfig)
    .build();

String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":
{\"XAttributeName\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}";
Parameter para5 = Parameter.builder()
    .name("parameter_step_1_vej_config")
    .value(para5JSON)
    .build();

System.out.println("parameter_step_1_vej_config:" + gson.toJson(jobConfig));
parameters.add(para1);
parameters.add(para2);
parameters.add(para3);
parameters.add(para4);
parameters.add(para5);

StartPipelineExecutionRequest pipelineExecutionRequest =
StartPipelineExecutionRequest.builder()
    .pipelineExecutionDescription("Created using Java SDK")
    .pipelineExecutionDisplayName(pipelineName + "-example-execution")
    .pipelineParameters(parameters)
    .pipelineName(pipelineName)
    .build();

StartPipelineExecutionResponse response =
sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
return response.pipelineExecutionArn();
}

public static void deleteEventSourceMapping(LambdaClient lambdaClient) {
    DeleteEventSourceMappingRequest eventSourceMappingRequest =
DeleteEventSourceMappingRequest.builder()
    .uuid(eventSourceMapping)
    .build();
```

```
        lambdaClient.deleteEventSourceMapping(eventSourceMappingRequest);
    }

    public static void deleteSagemakerRole(IamClient iam, String roleName) {
        String[] sagemakerRolePolicies = getSageMakerRolePolicies();
        try {
            for (String policy : sagemakerRolePolicies) {
                // First the policy needs to be detached.
                DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                    .policyArn(policy)
                    .roleName(roleName)
                    .build();

                iam.detachRolePolicy(rolePolicyRequest);
            }

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                .roleName(roleName)
                .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteLambdaRole(IamClient iam, String roleName) {
        String[] lambdaRolePolicies = getLambdaRolePolicies();
        try {
            for (String policy : lambdaRolePolicies) {
                // First the policy needs to be detached.
                DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                    .policyArn(policy)
                    .roleName(roleName)
                    .build();

                iam.detachRolePolicy(rolePolicyRequest);
            }
        }
```

```
// Delete the role.
DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
    .roleName(roleName)
    .build();

iam.deleteRole(roleRequest);
System.out.println("*** Successfully deleted " + roleName);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

// Delete the specific AWS Lambda function.
public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
    try {
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()
            .functionName(functionName)
            .build();

        awsLambda.deleteFunction(request);
        System.out.println("*** " + functionName + " was deleted");

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Delete the specific S3 bucket.
public static void deleteBucket(S3Client s3Client, String bucketName) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucketName)
        .build();
    s3Client.deleteBucket(deleteBucketRequest);
    System.out.println("*** " + bucketName + " was deleted.");
}

public static void listBucketObjects(S3Client s3, String bucketName) {
    try {
        ListObjectsRequest listObjects = ListObjectsRequest
```

```
        .builder()
        .bucket(bucketName)
        .build();

ListObjectsResponse res = s3.listObjects(listObjects);
List<S3Object> objects = res.contents();
for (S3Object myValue : objects) {
    System.out.print("\n The name of the key is " + myValue.key());
    deleteBucketObjects(s3, bucketName, myValue.key());
}

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void deleteBucketObjects(S3Client s3, String bucketName, String
objectName) {
    ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
    toDelete.add(ObjectIdentifier.builder()
        .key(objectName)
        .build());
    try {
        DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
            .bucket(bucketName)
            .delete(Delete.builder()
                .objects(toDelete).build())
            .build();

        s3.deleteObjects(dor);
        System.out.println("*** " + bucketName + " objects were deleted.");
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Delete the specific Amazon SQS queue.
public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
    try {
        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
            .queueName(queueName)
```

```
        .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
        DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
            .queueUrl(queueUrl)
            .build();

        sqsClient.deleteQueue(deleteQueueRequest);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {
    try {
        Map<String, String> metadata = new HashMap<>();
        metadata.put("x-amz-meta-myVal", "test");
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key("samplefiles/" + objectKey)
            .metadata(metadata)
            .build();

        s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
        System.out.println("Successfully placed " + objectKey + " into bucket "
+ bucketName);

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void setupBucket(S3Client s3Client, String bucketName) {
    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
    }
}
```



```
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Set up the SQS queue to use with the pipeline.
public static String setupQueue(SqsClient sqsClient, LambdaClient lambdaClient,
String queueName,
    String lambdaName) {
    System.out.println("Setting up queue named " + queueName);
    try {
        Map<QueueAttributeName, String> queueAtt = new HashMap<>();
        queueAtt.put(QueueAttributeName.DELAY_SECONDS, "5");
        queueAtt.put(QueueAttributeName.RECEIVE_MESSAGE_WAIT_TIME_SECONDS, "5");
        queueAtt.put(QueueAttributeName.VISIBILITY_TIMEOUT, "300");
        CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
            .queueName(queueName)
            .attributes(queueAtt)
            .build();

        sqsClient.createQueue(createQueueRequest);
        System.out.println("\nGet queue url");
        GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
        TimeUnit.SECONDS.sleep(15);

        connectLambda(sqsClient, lambdaClient, getQueueUrlResponse.queueUrl(),
lambdaName);
        System.out.println("Queue ready with Url " +
getQueueUrlResponse.queueUrl());
        return getQueueUrlResponse.queueUrl();
    }
}
```

```
    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
    return "";
}

// Connect the queue to the Lambda function as an event source.
public static void connectLambda(SqsClient sqsClient, LambdaClient lambdaClient,
String queueUrl,
    String lambdaName) {
    System.out.println("Connecting the Lambda function and queue for the
pipeline.");
    String queueArn = "";

    // Specify the attributes to retrieve.
    List<QueueAttributeName> atts = new ArrayList<>();
    atts.add(QueueAttributeName.QUEUE_ARN);
    GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
        .queueUrl(queueUrl)
        .attributeNames(atts)
        .build();

    GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
    Map<String, String> queueAtts = response.attributesAsStrings();
    for (Map.Entry<String, String> queueAtt : queueAtts.entrySet()) {
        System.out.println("Key = " + queueAtt.getKey() + ", Value = " +
queueAtt.getValue());
        queueArn = queueAtt.getValue();
    }

    CreateEventSourceMappingRequest eventSourceMappingRequest =
CreateEventSourceMappingRequest.builder()
        .eventSourceArn(queueArn)
        .functionName(lambdaName)
        .build();

    CreateEventSourceMappingResponse response1 =
lambdaClient.createEventSourceMapping(eventSourceMappingRequest);
    eventSourceMapping = response1.uuid();
}
```

```
        System.out.println("The mapping between the event source and Lambda function
was successful");
    }

    // Create an AWS Lambda function.
    public static String createLambdaFunction(LambdaClient awsLambda, String
functionName, String filePath, String role,
String handler) {
    try {
        LambdaWaiter waiter = awsLambda.waiter();
        InputStream is = new FileInputStream(filePath);
        SdkBytes fileToUpload = SdkBytes.fromInputStream(is);
        FunctionCode code = FunctionCode.builder()
            .zipFile(fileToUpload)
            .build();

        CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
            .functionName(functionName)
            .description("SageMaker example function.")
            .code(code)
            .handler(handler)
            .runtime(Runtime.JAVA11)
            .timeout(200)
            .memorySize(1024)
            .role(role)
            .build();

        // Create a Lambda function using a waiter.
        CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
        GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();

        WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("The function ARN is " +
functionResponse.functionArn());
        return functionResponse.functionArn();

    } catch (LambdaException | FileNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
    return "";
}

public static String createSageMakerRole(IamClient iam, String roleName) {
    String[] sageMakerRolePolicies = getSageMakerRolePolicies();
    System.out.println("Creating a role to use with SageMaker.");
    String assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\"," +
        "\"sagemaker-geospatial.amazonaws.com\"," +
        "\"lambda.amazonaws.com\"," +
        "\"s3.amazonaws.com\"" +
        "]" +
        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
        "}]}" +
        ";

    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(roleName)
            .assumeRolePolicyDocument(assumeRolePolicy)
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse roleResult = iam.createRole(request);

        // Attach the policies to the role.
        for (String policy : sageMakerRolePolicies) {
            AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(policy)
                .build();

            iam.attachRolePolicy(attachRequest);
        }

        // Allow time for the role to be ready.
        TimeUnit.SECONDS.sleep(15);
    }
}
```

```

        System.out.println("Role ready with ARN " + roleResult.role().arn());
        return roleResult.role().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
    return "";
}

private static String createLambdaRole(IamClient iam, String roleName) {
    String[] lambdaRolePolicies = getLambdaRolePolicies();
    String assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\"," +
        "\"sagemaker-geospatial.amazonaws.com\"," +
        "\"lambda.amazonaws.com\"," +
        "\"s3.amazonaws.com\"" +
        "]" +
        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
        "}]}" +
        "}";

    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(roleName)
            .assumeRolePolicyDocument(assumeRolePolicy)
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse roleResult = iam.createRole(request);

        // Attach the policies to the role.
        for (String policy : lambdaRolePolicies) {
            AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                .roleName(roleName)

```

```
        .policyArn(policy)
        .build();

    iam.attachRolePolicy(attachRequest);
}

// Allow time for the role to be ready.
TimeUnit.SECONDS.sleep(15);
System.out.println("Role ready with ARN " + roleResult.role().arn());
return roleResult.role().arn();

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());

} catch (InterruptedException e) {
    throw new RuntimeException(e);
}
return "";
}

public static String checkFunction(LambdaClient lambdaClient, String
functionName, String filePath, String role,
    String handler) {
    System.out.println("Create an AWS Lambda function used in this workflow.");
    String functionArn;
    try {
        // Does this function already exist.
        GetFunctionRequest functionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();

        GetFunctionResponse response =
lambdaClient.getFunction(functionRequest);
        functionArn = response.configuration().functionArn();

    } catch (LambdaException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        functionArn = createLambdaFunction(lambdaClient, functionName, filePath,
role, handler);
    }
    return functionArn;
}

// Check to see if the specific S3 bucket exists. If the S3 bucket exists, this
```

```
// method returns true.
public static boolean checkBucket(S3Client s3, String bucketName) {
    try {
        HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3.headBucket(headBucketRequest);
        System.out.println(bucketName + " exists");
        return true;

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    return false;
}

// Checks to see if the Amazon SQS queue exists. If not, this method creates a
// new queue
// and returns the ARN value.
public static String checkQueue(SqsClient sqsClient, LambdaClient lambdaClient,
String queueName,
    String lambdaName) {
    System.out.println("Creating a queue for this use case.");
    String queueUrl;
    try {
        GetQueueUrlRequest request = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        GetQueueUrlResponse response = sqsClient.getQueueUrl(request);
        queueUrl = response.queueUrl();
        System.out.println(queueUrl);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        queueUrl = setupQueue(sqsClient, lambdaClient, queueName, lambdaName);
    }
    return queueUrl;
}

// Checks to see if the Lambda role exists. If not, this method creates it.
public static String checkLambdaRole(IamClient iam, String roleName) {
    System.out.println("Creating a role to for AWS Lambda to use.");
```

```
String roleArn;
try {
    GetRoleRequest roleRequest = GetRoleRequest.builder()
        .roleName(roleName)
        .build();

    GetRoleResponse response = iam.getRole(roleRequest);
    roleArn = response.role().arn();
    System.out.println(roleArn);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    roleArn = createLambdaRole(iam, roleName);
}
return roleArn;
}

// Checks to see if the SageMaker role exists. If not, this method creates it.
public static String checkSageMakerRole(IamClient iam, String roleName) {
    System.out.println("Creating a role for AWS SageMaker to use.");
    String roleArn;
    try {
        GetRoleRequest roleRequest = GetRoleRequest.builder()
            .roleName(roleName)
            .build();

        GetRoleResponse response = iam.getRole(roleRequest);
        roleArn = response.role().arn();
        System.out.println(roleArn);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        roleArn = createSageMakerRole(iam, roleName);
    }
    return roleArn;
}

private static String[] getSageMakerRolePolicies() {
    String[] sageMakerRolePolicies = new String[3];
    sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess";
    sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/" +
    "AmazonSageMakerGeospatialFullAccess";
    sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
}
```



```
        return sageMakerRolePolicies;
    }

    private static String[] getLambdaRolePolicies() {
        String[] lambdaRolePolicies = new String[5];
        lambdaRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess";
        lambdaRolePolicies[1] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
        lambdaRolePolicies[2] = "arn:aws:iam::aws:policy/service-role/" +
"AmazonSageMakerGeospatialFullAccess";
        lambdaRolePolicies[3] = "arn:aws:iam::aws:policy/service-role/"
            + "AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy";
        lambdaRolePolicies[4] = "arn:aws:iam::aws:policy/service-role/" +
"AWSLambdaSQSQueueExecutionRole";
        return lambdaRolePolicies;
    }
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [CreatePipeline](#)
  - [DeletePipeline](#)
  - [DescribePipelineExecution](#)
  - [StartPipelineExecution](#)
  - [UpdatePipeline](#)

## Secrets-Manager-Beispiele mit SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Secrets Manager verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

## Themen

- [Aktionen](#)

## Aktionen

### Ein Secret erstellen

Das folgende Codebeispiel zeigt, wie Sie ein Secrets-Manager-Secret erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.CreateSecretRequest;
import software.amazon.awssdk.services.secretsmanager.model.CreateSecretResponse;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateSecret {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                    <secretName> <secretValue>\s

                Where:
                    secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
```

```
        secretValue - The secret value.\s
        """);

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String secretName = args[0];
    String secretValue = args[1];
    Region region = Region.US_EAST_1;
    SecretsManagerClient secretsClient = SecretsManagerClient.builder()
        .region(region)
        .build();

    String secretARN = createNewSecret(secretsClient, secretName, secretValue);
    System.out.println("The secret ARN is " + secretARN);
    secretsClient.close();
}

public static String createNewSecret(SecretsManagerClient secretsClient, String
secretName, String secretValue) {
    try {
        CreateSecretRequest secretRequest = CreateSecretRequest.builder()
            .name(secretName)
            .description("This secret was created by the AWS Secret Manager
Java API")
            .secretString(secretValue)
            .build();

        CreateSecretResponse secretResponse =
secretsClient.createSecret(secretRequest);
        return secretResponse.arn();

    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Weitere API-Informationen finden Sie unter [CreateSecret](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen eines Secrets

Das folgende Codebeispiel zeigt, wie Sie ein Secrets-Manager-Secret löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.DeleteSecretRequest;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteSecret {
    public static void main(String[] args) {

        final String usage = ""

                Usage:
                    <secretName>\s

                Where:
                    secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
                """;

        if (args.length != 1) {
```

```
        System.out.println(usage);
        System.exit(1);
    }

    String secretName = args[0];
    Region region = Region.US_EAST_1;
    SecretsManagerClient secretsClient = SecretsManagerClient.builder()
        .region(region)
        .build();

    deleteSpecificSecret(secretsClient, secretName);
    secretsClient.close();
}

public static void deleteSpecificSecret(SecretsManagerClient secretsClient,
String secretName) {
    try {
        DeleteSecretRequest secretRequest = DeleteSecretRequest.builder()
            .secretId(secretName)
            .build();

        secretsClient.deleteSecret(secretRequest);
        System.out.println(secretName + " is deleted.");

    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [DeleteSecret](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beschreiben eines Secrets

Das folgende Codebeispiel zeigt, wie Sie ein Secrets-Manager-Secret beschreiben.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.DescribeSecretRequest;
import software.amazon.awssdk.services.secretsmanager.model.DescribeSecretResponse;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeSecret {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <secretName>\s

                Where:
                secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String secretName = args[0];
    Region region = Region.US_EAST_1;
    SecretsManagerClient secretsClient = SecretsManagerClient.builder()
        .region(region)
        .build();

    describeGivenSecret(secretsClient, secretName);
    secretsClient.close();
}

public static void describeGivenSecret(SecretsManagerClient secretsClient,
String secretName) {
    try {
        DescribeSecretRequest secretRequest = DescribeSecretRequest.builder()
            .secretId(secretName)
            .build();

        DescribeSecretResponse secretResponse =
secretsClient.describeSecret(secretRequest);
        Instant lastChangedDate = secretResponse.lastChangedDate();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(lastChangedDate);
        System.out.println("The date of the last change to " +
secretResponse.name() + " is " + lastChangedDate);

    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [DescribeSecret](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Abrufen eines Secret-Werts

Das folgende Codebeispiel veranschaulicht, wie Sie einen Secrets-Manager-Geheimniswert abrufen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * We recommend that you cache your secret values by using client-side caching.
 *
 * Caching secrets improves speed and reduces your costs. For more information,
 * see the following documentation topic:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/retrieving-
 * secrets.html
 */
public class GetSecretValue {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <secretName>\s

                Where:
```



```
        secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
        """";

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String secretName = args[0];
    Region region = Region.US_EAST_1;
    SecretsManagerClient secretsClient = SecretsManagerClient.builder()
        .region(region)
        .build();

    getValue(secretsClient, secretName);
    secretsClient.close();
}

public static void getValue(SecretsManagerClient secretsClient, String
secretName) {
    try {
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
            .secretId(secretName)
            .build();

        GetSecretValueResponse valueResponse =
secretsClient.getSecretValue(valueRequest);
        String secret = valueResponse.secretString();
        System.out.println(secret);

    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [GetSecretValue](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Secrets

Das folgende Codebeispiel zeigt, wie Sie Secrets-Manager-Secrets auflisten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.ListSecretsResponse;
import software.amazon.awssdk.services.secretsmanager.model.SecretListEntry;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListSecrets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SecretsManagerClient secretsClient = SecretsManagerClient.builder()
            .region(region)
            .build();

        listAllSecrets(secretsClient);
        secretsClient.close();
    }

    public static void listAllSecrets(SecretsManagerClient secretsClient) {
        try {
            ListSecretsResponse secretsResponse = secretsClient.listSecrets();
            List<SecretListEntry> secrets = secretsResponse.secretList();
            for (SecretListEntry secret : secrets) {
```

```
        System.out.println("The secret name is " + secret.name());
        System.out.println("The secret description is " +
secret.description());
    }

    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [ListSecrets](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Ändert die Details eines Secrets

Das folgende Codebeispiel zeigt, wie das Secret geändert wird.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;
import software.amazon.awssdk.services.secretsmanager.model.UpdateSecretRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateSecret {
```

```
public static void main(String[] args) {
    final String usage = ""

        Usage:
            <secretName> <secretValue>

        Where:
            secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
            secretValue - The secret value that is updated.\s
        """;

    if (args.length < 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String secretName = args[0];
    String secretValue = args[1];
    Region region = Region.US_EAST_1;
    SecretsManagerClient secretsClient = SecretsManagerClient.builder()
        .region(region)
        .build();

    updateMySecret(secretsClient, secretName, secretValue);
    secretsClient.close();
}

public static void updateMySecret(SecretsManagerClient secretsClient, String
secretName, String secretValue) {
    try {
        UpdateSecretRequest secretRequest = UpdateSecretRequest.builder()
            .secretId(secretName)
            .secretString(secretValue)
            .build();

        secretsClient.updateSecret(secretRequest);

    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [UpdateSecret](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Einfügen eines Werts in ein Secret

Das folgende Codebeispiel zeigt, wie Sie einen Wert in ein Secrets-Manager-Secret einfügen.

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.PutSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutSecret {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <secretName> <secretValue>

                Where:
                secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
                secretValue - The text to encrypt and store in the new version
of the secret.\s
    }
}
```

```
        """);

    if (args.length < 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String secretName = args[0];
    String secretValue = args[1];
    Region region = Region.US_EAST_1;
    SecretsManagerClient secretsClient = SecretsManagerClient.builder()
        .region(region)
        .build();

    putSecret(secretsClient, secretName, secretValue);
    secretsClient.close();
}

public static void putSecret(SecretsManagerClient secretsClient, String
secretName, String secretValue) {
    try {
        PutSecretValueRequest secretRequest = PutSecretValueRequest.builder()
            .secretId(secretName)
            .secretString(secretValue)
            .build();

        secretsClient.putSecretValue(secretRequest);
        System.out.println("A new version was created.");

    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [PutSecretValue](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Amazon SES-Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon SES verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)

### Aktionen

#### Auflisten von E-Mail-Vorlagen

Das folgende Code-Beispiel zeigt, wie Sie Amazon-SES-E-Mail-Vorlagen auflisten.

SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesRequest;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesResponse;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;

public class ListTemplates {
```

```
public static void main(String[] args) {
    Region region = Region.US_EAST_1;
    SesV2Client sesv2Client = SesV2Client.builder()
        .region(region)
        .build();

    listAllTemplates(sesv2Client);
}

public static void listAllTemplates(SesV2Client sesv2Client) {
    try {
        ListEmailTemplatesRequest templatesRequest =
ListEmailTemplatesRequest.builder()
            .pageSize(1)
            .build();

        ListEmailTemplatesResponse response =
sesv2Client.listEmailTemplates(templatesRequest);
        response.templatesMetadata()
            .forEach(template -> System.out.println("Template name: " +
template.templateName()));

    } catch (SesV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [List Templates](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Identitäten

Das folgende Codebeispiel zeigt, wie Sie Amazon SES-Identitäten auflisten.



## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.ListIdentitiesResponse;
import software.amazon.awssdk.services.ses.model.SesException;
import java.io.IOException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListIdentities {

    public static void main(String[] args) throws IOException {
        Region region = Region.US_WEST_2;
        SesClient client = SesClient.builder()
            .region(region)
            .build();

        listSESIIdentities(client);
    }

    public static void listSESIIdentities(SesClient client) {
        try {
            ListIdentitiesResponse identitiesResponse = client.listIdentities();
            List<String> identities = identitiesResponse.identities();
            for (String identity : identities) {
                System.out.println("The identity is " + identity);
            }
        }
    }
}
```

```
        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [ListIdentities](#) in der APIAWS SDK for Java 2.x - Referenz für .

## E-Mail senden

Das folgende Codebeispiel zeigt, wie Sie E-Mails mit Amazon SES senden.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.Content;
import software.amazon.awssdk.services.ses.model.Destination;
import software.amazon.awssdk.services.ses.model.Message;
import software.amazon.awssdk.services.ses.model.Body;
import software.amazon.awssdk.services.ses.model.SendEmailRequest;
import software.amazon.awssdk.services.ses.model.SesException;

import javax.mail.MessagingException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class SendMessageEmailRequest {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <sender> <recipient> <subject>\s

            Where:
                sender - An email address that represents the sender.\s
                recipient - An email address that represents the recipient.\s
                subject - The subject line.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];

        Region region = Region.US_EAST_1;
        SesClient client = SesClient.builder()
            .region(region)
            .build();

        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
            + "<p> See the list of customers.</p>" + "</body>" + "</html>";

        try {
            send(client, sender, recipient, subject, bodyHTML);
            client.close();
            System.out.println("Done");
        } catch (MessagingException e) {
            e.printStackTrace();
        }
    }

    public static void send(SesClient client,
        String sender,
        String recipient,
```

```
String subject,
String bodyHTML) throws MessagingException {

    Destination destination = Destination.builder()
        .toAddresses(recipient)
        .build();

    Content content = Content.builder()
        .data(bodyHTML)
        .build();

    Content sub = Content.builder()
        .data(subject)
        .build();

    Body body = Body.builder()
        .html(content)
        .build();

    Message msg = Message.builder()
        .subject(sub)
        .body(body)
        .build();

    SendEmailRequest emailRequest = SendEmailRequest.builder()
        .destination(destination)
        .message(msg)
        .source(sender)
        .build();

    try {
        System.out.println("Attempting to send an email through Amazon SES " +
"using the AWS SDK for Java...");
        client.sendEmail(emailRequest);

    } catch (SesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
```

```
import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.mail.internet.MimeBodyPart;
import javax.mail.util.ByteArrayDataSource;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.file.Files;
import java.util.Properties;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.ses.model.SendRawEmailRequest;
import software.amazon.awssdk.services.ses.model.RawMessage;
import software.amazon.awssdk.services.ses.model.SesException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SendMessageAttachment {
    public static void main(String[] args) throws IOException {
        final String usage = ""

            Usage:
                <sender> <recipient> <subject> <fileLocation>\s

            Where:
                sender - An email address that represents the sender.\s
                recipient - An email address that represents the recipient.\s
                subject - The subject line.\s
                fileLocation - The location of a Microsoft Excel file to use as
                an attachment (C:/AWS/customers.xls).\s
            """;
```

```
    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String sender = args[0];
    String recipient = args[1];
    String subject = args[2];
    String fileLocation = args[3];

    // The email body for recipients with non-HTML email clients.
    String bodyText = "Hello,\r\n" + "Please see the attached file for a list "
        + "of customers to contact.";

    // The HTML body of the email.
    String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
        + "<p>Please see the attached file for a " + "list of customers to
contact.</p>" + "</body>"
        + "</html>";

    Region region = Region.US_WEST_2;
    SesClient client = SesClient.builder()
        .region(region)
        .build();

    try {
        sendemailAttachment(client, sender, recipient, subject, bodyText,
bodyHTML, fileLocation);
        client.close();
        System.out.println("Done");
    } catch (IOException | MessagingException e) {
        e.printStackTrace();
    }
}

public static void sendemailAttachment(SesClient client,
    String sender,
    String recipient,
    String subject,
    String bodyText,
    String bodyHTML,
```

```
String fileLocation) throws AddressException, MessagingException,
IOException {

    java.io.File theFile = new java.io.File(fileLocation);
    byte[] fileContent = Files.readAllBytes(theFile.toPath());

    Session session = Session.getDefaultInstance(new Properties());

    // Create a new MimeMessage object.
    MimeMessage message = new MimeMessage(session);

    // Add subject, from and to lines.
    message.setSubject(subject, "UTF-8");
    message.setFrom(new InternetAddress(sender));
    message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(recipient));

    // Create a multipart/alternative child container.
    MimeMultipart msgBody = new MimeMultipart("alternative");

    // Create a wrapper for the HTML and text parts.
    MimeBodyPart wrap = new MimeBodyPart();

    // Define the text part.
    MimeBodyPart textPart = new MimeBodyPart();
    textPart.setContent(bodyText, "text/plain; charset=UTF-8");

    // Define the HTML part.
    MimeBodyPart htmlPart = new MimeBodyPart();
    htmlPart.setContent(bodyHTML, "text/html; charset=UTF-8");

    // Add the text and HTML parts to the child container.
    msgBody.addBodyPart(textPart);
    msgBody.addBodyPart(htmlPart);

    // Add the child container to the wrapper object.
    wrap.setContent(msgBody);

    // Create a multipart/mixed parent container.
    MimeMultipart msg = new MimeMultipart("mixed");

    // Add the parent container to the message.
    message.setContent(msg);
    msg.addBodyPart(wrap);
```

```
// Define the attachment.
MimeBodyPart att = new MimeBodyPart();
DataSource fds = new ByteArrayDataSource(fileContent,
    "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet");
att.setDataHandler(new DataHandler(fds));

String reportName = "WorkReport.xls";
att.setFileName(reportName);

// Add the attachment to the message.
msg.addBodyPart(att);

try {
    System.out.println("Attempting to send an email through Amazon SES " +
"using the AWS SDK for Java...");

    ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
    message.writeTo(outputStream);

    ByteBuffer buf = ByteBuffer.wrap(outputStream.toByteArray());

    byte[] arr = new byte[buf.remaining()];
    buf.get(arr);

    SdkBytes data = SdkBytes.fromByteArray(arr);
    RawMessage rawMessage = RawMessage.builder()
        .data(data)
        .build();

    SendRawEmailRequest rawEmailRequest = SendRawEmailRequest.builder()
        .rawMessage(rawMessage)
        .build();

    client.sendRawEmail(rawEmailRequest);

} catch (SesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.println("Email sent using SesClient with attachment");
}
```



- Weitere API-Informationen finden Sie unter [SendEmail](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Senden einer Vorlage-E-Mail

Das folgende Code-Beispiel zeigt, wie man mit Amazon SES eine Vorlage-E-Mail sendet.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.Template;

/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * Also, make sure that you create a template. See the following documentation
 * topic:
 *
 * https://docs.aws.amazon.com/ses/latest/dg/send-personalized-email-api.html
 */

public class SendEmailTemplate {
    public static void main(String[] args) {
        final String usage = ""
```

## Usage:

```
<template> <sender> <recipient>\s
```

## Where:

template - The name of the email template.

sender - An email address that represents the sender.\s

recipient - An email address that represents the recipient.\s

```
""";
```

```
if (args.length != 3) {
    System.out.println(usage);
    System.exit(1);
}

String templateName = args[0];
String sender = args[1];
String recipient = args[2];
Region region = Region.US_EAST_1;
SesV2Client sesv2Client = SesV2Client.builder()
    .region(region)
    .build();

send(sesv2Client, sender, recipient, templateName);
}

public static void send(SesV2Client client, String sender, String recipient,
String templateName) {
    Destination destination = Destination.builder()
        .toAddresses(recipient)
        .build();

    /*
     * Specify both name and favorite animal (favoriteanimal) in your code when
     * defining the Template object.
     * If you don't specify all the variables in the template, Amazon SES
doesn't
     * send the email.
     */
    Template myTemplate = Template.builder()
        .templateName(templateName)
        .templateData("{\n" +
            "  \"name\": \"Jason\"\n," +
            "  \"favoriteanimal\": \"Cat\"\n" +
```

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

    EmailContent emailContent = EmailContent.builder()
        .template(myTemplate)
        .build();

    SendEmailRequest emailRequest = SendEmailRequest.builder()
        .destination(destination)
        .content(emailContent)
        .fromEmailAddress(sender)
        .build();

    try {
        System.out.println("Attempting to send an email based on a template
using the AWS SDK for Java (v2)...");
        client.sendEmail(emailRequest);
        System.out.println("email based on a template was sent");

    } catch (SesV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [SendTemplatedEmail](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Amazon SES-API-v2-Beispiele mit SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon SES-API v2 verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)

## Aktionen

E-Mail senden

Das folgende Code-Beispiel zeigt, wie eine Amazon-SES-API-v2-E-Mail gesendet wird.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Sendet eine Nachricht.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Body;
import software.amazon.awssdk.services.sesv2.model.Content;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.Message;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;

/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class SendEmail {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <sender> <recipient> <subject>\s

            Where:
                sender - An email address that represents the
sender.\s
                recipient - An email address that represents the
recipient.\s
                subject - The subject line.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];

        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
            .region(region)
            .build();

        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" +
"<h1>Hello!</h1>"
            + "<p> See the list of customers.</p>" + "</body>" +
"</html>";

        send(sesv2Client, sender, recipient, subject, bodyHTML);
    }

    public static void send(SesV2Client client,
        String sender,
        String recipient,
        String subject,
        String bodyHTML) {
```

```
Destination destination = Destination.builder()
    .toAddresses(recipient)
    .build();

Content content = Content.builder()
    .data(bodyHTML)
    .build();

Content sub = Content.builder()
    .data(subject)
    .build();

Body body = Body.builder()
    .html(content)
    .build();

Message msg = Message.builder()
    .subject(sub)
    .body(body)
    .build();

EmailContent emailContent = EmailContent.builder()
    .simple(msg)
    .build();

SendEmailRequest emailRequest = SendEmailRequest.builder()
    .destination(destination)
    .content(emailContent)
    .fromEmailAddress(sender)
    .build();

try {
    System.out.println("Attempting to send an email through
Amazon SES "
        + "using the AWS SDK for Java...");
    client.sendEmail(emailRequest);
    System.out.println("email was sent");
} catch (SesV2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [SendEmail](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Amazon SNS-Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon SNS verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Erste Schritte

#### Hello Amazon SNS

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon SNS.

#### SDK für Java 2.x

##### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
package com.example.sns;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.paginators.ListTopicsIterable;
```

```
public class HelloSNS {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsIterable listTopics = snsClient.listTopicsPaginator();
            listTopics.stream()
                .flatMap(r -> r.topics().stream())
                .forEach(content -> System.out.println(" Topic ARN: " +
content.topicArn()));

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [ListTopics](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Themen

- [Aktionen](#)
- [Szenarien](#)
- [Serverless-Beispiele](#)

## Aktionen

### Hinzufügen von Tags zu einem Thema

Im folgenden Codebeispiel wird veranschaulicht, wie Sie einem Amazon-SNS-Thema Tags hinzufügen.



## SDK for Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.Tag;
import software.amazon.awssdk.services.sns.model.TagResourceRequest;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AddTags {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn>

            Where:
                topicArn - The ARN of the topic to which tags are added.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    addTopicTags(snsClient, topicArn);
    snsClient.close();
}

public static void addTopicTags(SnsClient snsClient, String topicArn) {
    try {
        Tag tag = Tag.builder()
            .key("Team")
            .value("Development")
            .build();

        Tag tag2 = Tag.builder()
            .key("Environment")
            .value("Gamma")
            .build();

        List<Tag> tagList = new ArrayList<>();
        tagList.add(tag);
        tagList.add(tag2);

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .resourceArn(topicArn)
            .tags(tagList)
            .build();

        snsClient.tagResource(tagResourceRequest);
        System.out.println("Tags have been added to " + topicArn);

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [TagResource](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Überprüfen, ob eine Telefonnummer den Empfang abgemeldet ist

Das folgende Codebeispiel zeigt, wie Sie überprüfen können, ob eine Telefonnummer den Empfang von Amazon SNS-Nachrichten deaktiviert hat.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
    software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutRequest;
import
    software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CheckOptOut {
    public static void main(String[] args) {

        final String usage = ""

            Usage:    <phoneNumber>

            Where:
                phoneNumber - The mobile phone number to look up (for example,
                +1XXX5550100).

            """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String phoneNumber = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        checkPhone(snsClient, phoneNumber);
        snsClient.close();
    }

    public static void checkPhone(SnsClient snsClient, String phoneNumber) {
        try {
            CheckIfPhoneNumberIsOptedOutRequest request =
                CheckIfPhoneNumberIsOptedOutRequest.builder()
                    .phoneNumber(phoneNumber)
                    .build();

            CheckIfPhoneNumberIsOptedOutResponse result =
                snsClient.checkIfPhoneNumberIsOptedOut(request);
            System.out.println(
                result.isOptedOut() + "Phone Number " + phoneNumber + " has
                Opted Out of receiving sns messages." +
                "\n\nStatus was " +
                result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [CheckIfPhoneNumberIsOptedOut](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Bestätigen, dass der Eigentümer eines Endpunkts Nachrichten empfangen möchte

Das folgende Codebeispiel zeigt, wie bestätigt wird, dass der Besitzer eines Endpunkts Amazon SNS-Nachrichten empfangen möchte, indem das Token validiert wird, das von einer früheren Abonnementaktion an den Endpunkt gesendet wurde.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionRequest;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ConfirmSubscription {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <subscriptionToken> <topicArn>

                Where:
                    subscriptionToken - A short-lived token sent to an endpoint
during the Subscribe action.
                    topicArn - The ARN of the topic.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String subscriptionToken = args[0];
    String topicArn = args[1];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    confirmSub(snsClient, subscriptionToken, topicArn);
    snsClient.close();
}

public static void confirmSub(SnsClient snsClient, String subscriptionToken,
String topicArn) {
    try {
        ConfirmSubscriptionRequest request =
ConfirmSubscriptionRequest.builder()
            .token(subscriptionToken)
            .topicArn(topicArn)
            .build();

        ConfirmSubscriptionResponse result =
snsClient.confirmSubscription(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode() + "\n\nSubscription Arn: \n\n"
            + result.subscriptionArn());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [ConfirmSubscription](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erstellen eines Themas

Das folgende Codebeispiel zeigt, wie Sie ein Amazon SNS-Thema erstellen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicName>

            Where:
                topicName - The name of the topic to create (for example,
mytopic).

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    String arnVal = createSNSTopic(snsClient, topicName);
    System.out.println("The topic ARN is" + arnVal);
    snsClient.close();
}

public static String createSNSTopic(SnsClient snsClient, String topicName) {
    CreateTopicResponse result;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .build();

        result = snsClient.createTopic(request);
        return result.topicArn();

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Weitere API-Informationen finden Sie unter [CreateTopic](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen von Abonnements

Das folgende Codebeispiel zeigt, wie Sie ein Amazon SNS-Abonnement löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.



```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class Unsubscribe {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <subscriptionArn>

            Where:
                subscriptionArn - The ARN of the subscription to delete.
            "";

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        unSub(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                .subscriptionArn(subscriptionArn)
                .build();
        }
    }
}
```

```
        UnsubscribeResponse result = snsClient.unsubscribe(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode()
        + "\n\nSubscription was removed for " +
request.subscriptionArn());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Details zu API finden Sie unter [Abmelden](#) in der AWS SDK for Java 2.x -API-Referenz.

## Löschen eines Themas

Das folgende Codebeispiel zeigt, wie Sie ein Amazon SNS-Thema und alle Abonnements für dieses Thema löschen.

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */
```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DeleteTopic {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn>

            Where:
                topicArn - The ARN of the topic to delete.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                .topicArn(topicArn)
                .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteTopic](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Abrufen der Eigenschaften eines Themas

Das folgende Codebeispiel zeigt, wie Sie die Eigenschaften eines Amazon SNS-Themas abrufen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetTopicAttributes {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <topicArn>

                Where:
                    topicArn - The ARN of the topic to look up.
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        System.out.println("Getting attributes for a topic with name: " + topicArn);
        getSNSTopicAttributes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSTopicAttributes(SnsClient snsClient, String topicArn) {
        try {
            GetTopicAttributesRequest request = GetTopicAttributesRequest.builder()
                .topicArn(topicArn)
                .build();

            GetTopicAttributesResponse result =
snsClient.getTopicAttributes(request);
            System.out.println("\n\nStatus is " +
result.sdkHttpResponse().statusCode() + "\n\nAttributes: \n\n"
                + result.attributes());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [GetTopicAttributes](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Abrufen der Einstellungen für das Senden von SMS-Nachrichten

Das folgende Codebeispiel zeigt, wie Sie die Einstellungen für das Senden von Amazon SNS-SMS-Nachrichten abrufen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.Iterator;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetSMSAttributes {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <topicArn>

                Where:
                    topicArn - The ARN of the topic from which to retrieve
attributes.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```

```

        .region(Region.US_EAST_1)
        .build();

    getSNSAttrutes(snsClient, topicArn);
    snsClient.close();
}

public static void getSNSAttrutes(SnsClient snsClient, String topicArn) {
    try {
        GetSubscriptionAttributesRequest request =
        GetSubscriptionAttributesRequest.builder()
            .subscriptionArn(topicArn)
            .build();

        // Get the Subscription attributes
        GetSubscriptionAttributesResponse res =
        snsClient.getSubscriptionAttributes(request);
        Map<String, String> map = res.attributes();

        // Iterate through the map
        Iterator iter = map.entrySet().iterator();
        while (iter.hasNext()) {
            Map.Entry entry = (Map.Entry) iter.next();
            System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
            entry.getValue());
        }

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    System.out.println("\n\nStatus was good");
}
}

```

- Details zu API finden Sie unter [GetSMSAttributes](#) in der AWS SDK for Java 2.x -API-Referenz.

## Auflisten abgemeldeter Telefonnummern

Das folgende Codebeispiel zeigt, wie Telefonnummern aufgelistet werden, die vom Empfang von Amazon SNS-Nachrichten abgemeldet wurden.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutRequest;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListOptOut {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listOpts(snsClient);
        snsClient.close();
    }

    public static void listOpts(SnsClient snsClient) {
        try {
            ListPhoneNumbersOptedOutRequest request =
ListPhoneNumbersOptedOutRequest.builder().build();
            ListPhoneNumbersOptedOutResponse result =
snsClient.listPhoneNumbersOptedOut(request);
            System.out.println("Status is " + result.sdkHttpResponse().statusCode()
+ "\n\nPhone Numbers: \n\n"
                + result.phoneNumbers());
        }
    }
}
```



```
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [ListPhoneNumbersOptedOut](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Listen der Abonnenten eines Themas

Das folgende Codebeispiel zeigt, wie Sie die Liste der Abonnenten eines Amazon SNS-Themas abrufen.

### SDK für Java 2.x

#### Note

Auf [GitHub](#) gibt es mehr Beispiele. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsRequest;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListSubscriptions {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    listSNSSubscriptions(snsClient);
    snsClient.close();
}

public static void listSNSSubscriptions(SnsClient snsClient) {
    try {
        ListSubscriptionsRequest request = ListSubscriptionsRequest.builder()
            .build();

        ListSubscriptionsResponse result = snsClient.listSubscriptions(request);
        System.out.println(result.subscriptions());

    } catch (SnsException e) {

        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [ListSubscriptions](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Themen

Das folgende Codebeispiel zeigt, wie Sie Amazon SNS-Themen auflisten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
```

```
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTopics {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsRequest request = ListTopicsRequest.builder()
                .build();

            ListTopicsResponse result = snsClient.listTopics(request);
            System.out.println(
                "Status was " + result.sdkHttpResponse().statusCode() + "\n\n"
                + result.topics());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [ListTopics](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Veröffentlichen einer SMS-Nachricht

Das folgende Codebeispiel zeigt, wie SMS-Nachrichten mit Amazon SNS veröffentlicht werden.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <message> <phoneNumber>

                Where:
                    message - The message text to send.
                    phoneNumber - The mobile phone number to which a message is sent
(for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String message = args[0];
String phoneNumber = args[1];
SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .build();
pubTextSMS(snsClient, message, phoneNumber);
snsClient.close();
}

public static void pubTextSMS(SnsClient snsClient, String message, String
phoneNumber) {
    try {
        PublishRequest request = PublishRequest.builder()
            .message(message)
            .phoneNumber(phoneNumber)
            .build();

        PublishResponse result = snsClient.publish(request);
        System.out
            .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Details zu API finden Sie unter [Veröffentlichen](#) in der AWS SDK for Java 2.x -API-Referenz.

## Veröffentlichung für ein Thema

Das folgende Codebeispiel zeigt, wie Nachrichten in einem Amazon SNS-Thema veröffentlicht werden.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTopic {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <message> <topicArn>

                Where:
                    message - The message text to send.
                    topicArn - The ARN of the topic to publish.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
```

```
        .build();
        pubTopic(snsClient, message, topicArn);
        snsClient.close();
    }

    public static void pubTopic(SnsClient snsClient, String message, String
topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                .message(message)
                .topicArn(topicArn)
                .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Details zu API finden Sie unter [Veröffentlichen](#) in der AWS SDK for Java 2.x -API-Referenz.

## Festlegen einer Filterrichtlinie

Das folgende Codebeispiel zeigt, wie Sie eine Amazon SNS-Filterrichtlinie festlegen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
```

```
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UseMessageFilterPolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <subscriptionArn>

            Where:
                subscriptionArn - The ARN of a subscription.

            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        usePolicy(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
        try {
            SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
            // Add a filter policy attribute with a single value
            fp.addAttribute("store", "example_corp");
            fp.addAttribute("event", "order_placed");

            // Add a prefix attribute
```



```
fp.addAttributePrefix("customer_interests", "bas");

// Add an anything-but attribute
fp.addAttributeAnythingBut("customer_interests", "baseball");

// Add a filter policy attribute with a list of values
ArrayList<String> attributeValues = new ArrayList<>();
attributeValues.add("rugby");
attributeValues.add("soccer");
attributeValues.add("hockey");
fp.addAttribute("customer_interests", attributeValues);

// Add a numeric attribute
fp.addAttribute("price_usd", "=", 0);

// Add a numeric attribute with a range
fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

// Apply the filter policy attributes to an Amazon SNS subscription
fp.apply(snsClient, subscriptionArn);

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [SetSubscriptionAttributes](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Festlegen der Standardeinstellungen für das Senden von SMS-Nachrichten

Das folgende Codebeispiel zeigt, wie Sie die Standardeinstellungen für das Senden von SMS-Nachrichten mit Amazon SNS festlegen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String, String>
attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                .attributes(attributes)
                .build();
```

```
        SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
        System.out.println("Set default Attributes to " + attributes + ". Status
was "
            + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Details zu API finden Sie unter [SetSMSAttributes](#) in der AWS SDK for Java 2.x -API-Referenz.

## Festlegen von Themenattributen

Das folgende Codebeispiel zeigt, wie Sie Amazon SNS-Themenattribute festlegen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class SetTopicAttributes {

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <attribute> <topicArn> <value>

            Where:
                attribute - The attribute action to use. Valid parameters are:
Policy | DisplayName | DeliveryPolicy .
                topicArn - The ARN of the topic.\s
                value - The value for the attribute.
            """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String attribute = args[0];
        String topicArn = args[1];
        String value = args[2];

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        setTopAttr(snsClient, attribute, topicArn, value);
        snsClient.close();
    }

    public static void setTopAttr(SnsClient snsClient, String attribute, String
topicArn, String value) {
        try {
            SetTopicAttributesRequest request = SetTopicAttributesRequest.builder()
                .attributeName(attribute)
                .attributeValue(value)
                .topicArn(topicArn)
                .build();

            SetTopicAttributesResponse result =
snsClient.setTopicAttributes(request);
            System.out.println(
```

```

        "\n\nStatus was " + result.sdkHttpResponse().statusCode() + "\n\nTopic " + request.topicArn()
            + " updated " + request.attributeName() + " to " +
            request.attributeValue());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Weitere API-Informationen finden Sie unter [SetTopicAttributes](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Abonnieren eines Themas mit einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie Sie eine Lambda-Funktion abonnieren, damit sie Benachrichtigungen von einem Amazon SNS-Thema erhält.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */

```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class SubscribeLambda {

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <topicArn> <lambdaArn>

            Where:
                topicArn - The ARN of the topic to subscribe.
                lambdaArn - The ARN of an AWS Lambda function.
            "";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String lambdaArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String arnValue = subLambda(snsClient, topicArn, lambdaArn);
        System.out.println("Subscription ARN: " + arnValue);
        snsClient.close();
    }

    public static String subLambda(SnsClient snsClient, String topicArn, String
lambdaArn) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                .protocol("lambda")
                .endpoint(lambdaArn)
                .returnSubscriptionArn(true)
                .topicArn(topicArn)
                .build();

            SubscribeResponse result = snsClient.subscribe(request);
            return result.subscriptionArn();
        }
    }
}
```

```
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Details zu API finden Sie unter [Abonnieren](#) in der AWS SDK for Java 2.x -API-Referenz.

## Abonnieren eines Endpunkts für ein Thema

Das folgende Codebeispiel zeigt, wie Sie einen HTTP- oder HTTPS-Endpunkt abonnieren, damit er Benachrichtigungen von einem Amazon SNS-Thema erhält.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeHTTPS {
    public static void main(String[] args) {
        final String usage = ""
```

```
Usage:    <topicArn> <url>

Where:
  topicArn - The ARN of the topic to subscribe.
  url      - The HTTPS endpoint that you want to receive notifications.
""";

if (args.length < 2) {
    System.out.println(usage);
    System.exit(1);
}

String topicArn = args[0];
String url = args[1];
SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .build();

subHTTPS(snsClient, topicArn, url);
snsClient.close();
}

public static void subHTTPS(SnsClient snsClient, String topicArn, String url) {
    try {
        SubscribeRequest request = SubscribeRequest.builder()
            .protocol("https")
            .endpoint(url)
            .returnSubscriptionArn(true)
            .topicArn(topicArn)
            .build();

        SubscribeResponse result = snsClient.subscribe(request);
        System.out.println("Subscription ARN is " + result.subscriptionArn() +
            "\n\n Status is "
                + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```



- Details zu API finden Sie unter [Abonnieren](#) in der AWS SDK for Java 2.x -API-Referenz.

## Abonnieren einer E-Mail-Adresse für ein Thema

Das folgende Codebeispiel zeigt, wie Sie eine E-Mail-Adresse für ein Amazon SNS-Thema abonnieren.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeEmail {
    public static void main(String[] args) {
        final String usage = ""
            Usage:    <topicArn> <email>

            Where:
                topicArn - The ARN of the topic to subscribe.
                email - The email address to use.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String topicArn = args[0];
    String email = args[1];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    subEmail(snsClient, topicArn, email);
    snsClient.close();
}

public static void subEmail(SnsClient snsClient, String topicArn, String email)
{
    try {
        SubscribeRequest request = SubscribeRequest.builder()
            .protocol("email")
            .endpoint(email)
            .returnSubscriptionArn(true)
            .topicArn(topicArn)
            .build();

        SubscribeResponse result = snsClient.subscribe(request);
        System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n\n"
            + "Status is "
                + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Details zu API finden Sie unter [Abonnieren](#) in der AWS SDK for Java 2.x -API-Referenz.

## Szenarien

### Erstellen eines Plattformendpunkts für Push-Benachrichtigungen

Das folgende Code-Beispiel zeigt, wie man ein Plattformendpunkt für Amazon-SNS-Push-Benachrichtigungen erstellt.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = ""

            Usage:      <token> <platformApplicationArn>

            Where:
                token - The name of the FIFO topic.\s
                platformApplicationArn - The ARN value of platform application.
            You can get this value from the AWS Management Console.\s
            """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }

    public static void createEndpoint(SnsClient snsClient, String token, String
platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
CreatePlatformEndpointRequest.builder()
                .token(token)
                .platformApplicationArn(platformApplicationArn)
                .build();

            CreatePlatformEndpointResponse response =
snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
response.endpointArn());
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

## Erstellen und veröffentlichen zu einem FIFO-Thema

Die folgenden Code-Beispiele zeigen, wie man ein Amazon-SNS-Thema erstellt.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Dieses Beispiel

- erstellt ein Amazon-SNS-FIFO-Thema, zwei Amazon SQS-FIFO-Warteschlangen und eine Standard-Warteschlange.
- abonniert die Warteschlangen für das Thema und veröffentlicht eine Nachricht zu dem Thema.

Der [Test](#) überprüft den Eingang der Nachricht in jeder Warteschlange. Das [vollständige Beispiel](#) zeigt auch das Hinzufügen von Zugriffsrichtlinien und löscht die Ressourcen am Ende.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
<analyticsQueueName>\n\n" +
            "Where:\n" +
            "    fifoTopicName - The name of the FIFO topic that you want to
create. \n\n" +
            "    wholesaleQueueARN - The name of a SQS FIFO queue that will be
created for the wholesale consumer. \n\n"
            +
            "    retailQueueARN - The name of a SQS FIFO queue that will created
for the retail consumer. \n\n" +
            "    analyticsQueueARN - The name of a SQS standard queue that will
be created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
```

```
final String wholeSaleQueueName = args[1];
final String retailQueueName = args[2];
final String analyticsQueueName = args[3];

// For convenience, the QueueData class holds metadata about a queue: ARN,
URL,
// name and type.
List<QueueData> queues = List.of(
    new QueueData(wholeSaleQueueName, QueueType.FIFO),
    new QueueData(retailQueueName, QueueType.FIFO),
    new QueueData(analyticsQueueName, QueueType.Standard));

// Create queues.
createQueues(queues);

// Create a topic.
String topicARN = createFIFOTopic(fifoTopicName);

// Subscribe each queue to the topic.
subscribeQueues(queues, topicARN);

// Allow the newly created topic to send messages to the queues.
addAccessPolicyToQueuesFINAL(queues, topicARN);

// Publish a sample price update message with payload.
publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

// Clean up resources.
deleteSubscriptions(queues);
deleteQueues(queues);
deleteTopic(topicARN);
}

public static String createFIFOTopic(String topicName) {
    try {
        // Create a FIFO topic by using the SNS service client.
        Map<String, String> topicAttributes = Map.of(
            "FifoTopic", "true",
            "ContentBasedDeduplication", "false");

        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
            .name(topicName)
            .attributes(topicAttributes)
```

```
        .build();

        CreateTopicResponse response = snsClient.createTopic(topicRequest);
        String topicArn = response.topicArn();
        System.out.println("The topic ARN is" + topicArn);

        return topicArn;

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void subscribeQueues(List<QueueData> queues, String topicARN) {
    queues.forEach(queue -> {
        SubscribeRequest subscribeRequest = SubscribeRequest.builder()
            .topicArn(topicARN)
            .endpoint(queue.queueARN)
            .protocol("sqs")
            .build();

        // Subscribe to the endpoint by using the SNS service client.
        // Only Amazon SQS queues can receive notifications from an Amazon SNS
        FIFO
        // topic.
        SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
        System.out.println("The queue [" + queue.queueARN + "] subscribed to the
topic [" + topicARN + "]);
        queue.subscriptionARN = subscribeResponse.subscriptionArn();
    });
}

public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {
    try {
        // Create and publish a message that updates the wholesale price.
        String subject = "Price Update";
        String dedupId = UUID.randomUUID().toString();
        String attributeName = "business";
        String attributeValue = "wholesale";
```

```
MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
    .dataType("String")
    .stringValue(attributeValue)
    .build();

Map<String, MessageAttributeValue> attributes = new HashMap<>();
attributes.put(attributeName, msgAttValue);
PublishRequest pubRequest = PublishRequest.builder()
    .topicArn(topicArn)
    .subject(subject)
    .message(payload)
    .messageGroupId(groupId)
    .messageDeduplicationId(dedupId)
    .messageAttributes(attributes)
    .build();

final PublishResponse response = snsClient.publish(pubRequest);
System.out.println(response.messageId());
System.out.println(response.sequenceNumber());
System.out.println("Message was published to " + topicArn);

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [CreateTopic](#)
  - [Veröffentlichen](#)
  - [Abonnieren](#)

## Veröffentlichen einer SMS-Nachricht zu einem Thema

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Amazon-SNS-Thema.
- Verknüpfen Sie Telefonnummern mit dem Thema.



- Veröffentlichen Sie SMS-Nachrichten im Thema, damit alle abonnierten Telefonnummern die Nachricht gleichzeitig empfangen.

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Erstellen Sie ein Thema und geben Sie seinen ARN zurück.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <topicName>

                Where:
                    topicName - The name of the topic to create (for example,
mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```

    }

    String topicName = args[0];
    System.out.println("Creating a topic with name: " + topicName);
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    String arnVal = createSNSTopic(snsClient, topicName);
    System.out.println("The topic ARN is" + arnVal);
    snsClient.close();
}

public static String createSNSTopic(SnsClient snsClient, String topicName) {
    CreateTopicResponse result;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .build();

        result = snsClient.createTopic(request);
        return result.topicArn();

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
}

```

## Abonnieren eines Endpunkts für ein Thema.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.

```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class SubscribeTextSMS {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn> <phoneNumber>

            Where:
                topicArn - The ARN of the topic to subscribe.
                phoneNumber - A mobile phone number that receives notifications
(for example, +1XXX5550100).
            """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        subTextSNS(snsClient, topicArn, phoneNumber);
        snsClient.close();
    }

    public static void subTextSNS(SnsClient snsClient, String topicArn, String
phoneNumber) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                .protocol("sms")
                .endpoint(phoneNumber)
                .returnSubscriptionArn(true)
                .topicArn(topicArn)
                .build();

            SubscribeResponse result = snsClient.subscribe(request);
```

```

        System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
        + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

Legen Sie Attribute für die Nachricht fest, z. B. die ID des Senders, den Höchstpreis und seinen Typ. Nachrichtenattribute sind optional.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }
}

```

```

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String, String>
attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                .attributes(attributes)
                .build();

            SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ". Status
was "
                + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

Veröffentlichen einer Nachricht für ein Thema. Die Nachricht wird an jeden Teilnehmer gesendet.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <message> <phoneNumber>

            Where:

```

```
        message - The message text to send.
        phoneNumber - The mobile phone number to which a message is sent
(for example, +1XXX5550100).\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String message = args[0];
    String phoneNumber = args[1];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();
    pubTextSMS(snsClient, message, phoneNumber);
    snsClient.close();
}

public static void pubTextSMS(SnsClient snsClient, String message, String
phoneNumber) {
    try {
        PublishRequest request = PublishRequest.builder()
            .message(message)
            .phoneNumber(phoneNumber)
            .build();

        PublishResponse result = snsClient.publish(request);
        System.out
            .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

## Serverless-Beispiele

### Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten von einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

### Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Java.

```
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;

import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
    }
}
```

```
    }
    }
    return Boolean.TRUE;
}

public void processRecord(SNSRecord record) {
    try {
        String message = record.getSNS().getMessage();
        logger.log("message: " + message);
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
}
```

## Amazon SQS-Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon SQS verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Erste Schritte

#### Hallo Amazon SQS

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon SQS .



## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.SqsException;
import software.amazon.awssdk.services.sqs.paginators.ListQueuesIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloSQS {
    public static void main(String[] args) {
        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_WEST_2)
            .build();

        listQueues(sqsClient);
        sqsClient.close();
    }

    public static void listQueues(SqsClient sqsClient) {
        try {
            ListQueuesIterable listQueues = sqsClient.listQueuesPaginator();
            listQueues.stream()
                .flatMap(r -> r.queueUrls().stream())
                .forEach(content -> System.out.println(" Queue URL: " +
                    content.toLowerCase()));
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
    }  
  }  
}
```

- Weitere API-Informationen finden Sie unter [ListQueues](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Themen

- [Aktionen](#)
- [Szenarien](#)
- [Serverless-Beispiele](#)

## Aktionen

### Erstellen einer Warteschlange

Das folgende Codebeispiel zeigt, wie Sie eine Amazon SQS-Warteschlange erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.sqs.SqsClient;  
import software.amazon.awssdk.services.sqs.model.ChangeMessageVisibilityRequest;  
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;  
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;  
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;  
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;  
import software.amazon.awssdk.services.sqs.model.ListQueuesRequest;  
import software.amazon.awssdk.services.sqs.model.ListQueuesResponse;  
import software.amazon.awssdk.services.sqs.model.Message;  
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;  
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequest;
```

```
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SQSExample {
    public static void main(String[] args) {
        String queueName = "queue" + System.currentTimeMillis();
        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_WEST_2)
            .build();

        // Perform various tasks on the Amazon SQS queue.
        String queueUrl = createQueue(sqsClient, queueName);
        listQueues(sqsClient);
        listQueuesFilter(sqsClient, queueUrl);
        List<Message> messages = receiveMessages(sqsClient, queueUrl);
        sendBatchMessages(sqsClient, queueUrl);
        changeMessages(sqsClient, queueUrl, messages);
        deleteMessages(sqsClient, queueUrl, messages);
        sqsClient.close();
    }

    public static String createQueue(SqsClient sqsClient, String queueName) {
        try {
            System.out.println("\nCreate Queue");

            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                .queueName(queueName)
                .build();

            sqsClient.createQueue(createQueueRequest);

            System.out.println("\nGet queue url");

            GetQueueUrlResponse getQueueUrlResponse = sqsClient
```

```
.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
    return getQueueUrlResponse.queueUrl();

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void listQueues(SqsClient sqsClient) {

    System.out.println("\nList Queues");
    String prefix = "que";

    try {
        ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
        ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
        for (String url : listQueuesResponse.queueUrls()) {
            System.out.println(url);
        }

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void listQueuesFilter(SqsClient sqsClient, String queueUrl) {
    // List queues with filters
    String namePrefix = "queue";
    ListQueuesRequest filterListRequest = ListQueuesRequest.builder()
        .queueNamePrefix(namePrefix)
        .build();

    ListQueuesResponse listQueuesFilteredResponse =
sqsClient.listQueues(filterListRequest);
    System.out.println("Queue URLs with prefix: " + namePrefix);
    for (String url : listQueuesFilteredResponse.queueUrls()) {
        System.out.println(url);
    }
}
```

```
System.out.println("\nSend message");
try {
    sqsClient.sendMessage(SendMessageRequest.builder()
        .queueUrl(queueUrl)
        .messageBody("Hello world!")
        .delaySeconds(10)
        .build());

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void sendBatchMessages(SqsClient sqsClient, String queueUrl) {

    System.out.println("\nSend multiple messages");
    try {
        SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
            .queueUrl(queueUrl)

.entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from
msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)

                .build())
            .build();
        sqsClient.sendMessageBatch(sendMessageBatchRequest);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl) {

    System.out.println("\nReceive messages");
    try {
```

```
        ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
    .queueUrl(queueUrl)
    .numberOfMessages(5)
    .build();
        return sqsClient.receiveMessage(receiveMessageRequest).messages();

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static void changeMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {

    System.out.println("\nChange Message Visibility");
    try {

        for (Message message : messages) {
            ChangeMessageVisibilityRequest req =
ChangeMessageVisibilityRequest.builder()
                .queueUrl(queueUrl)
                .receiptHandle(message.receiptHandle())
                .visibilityTimeout(100)
                .build();
            sqsClient.changeMessageVisibility(req);
        }

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
    System.out.println("\nDelete Messages");

    try {
        for (Message message : messages) {
            DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
```

```
        .queueUrl(queueUrl)
        .receiptHandle(message.receiptHandle())
        .build();
    sqsClient.deleteMessage(deleteMessageRequest);
    }
} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [CreateQueue](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen einer Nachricht aus einer Warteschlange

Das folgende Codebeispiel zeigt, wie Sie eine Nachricht aus einer Amazon SQS-Warteschlange löschen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
    try {
        for (Message message : messages) {
            DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                .queueUrl(queueUrl)
                .receiptHandle(message.receiptHandle())
                .build();
            sqsClient.deleteMessage(deleteMessageRequest);
        }
    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
```

```
}
```

- Weitere API-Informationen finden Sie unter [DeleteMessage](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen einer Warteschlange

Das folgende Codebeispiel zeigt, wie Sie eine Amazon SQS-Warteschlange löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteQueue {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <queueName>

                Where:
                queueName - The name of the Amazon SQS queue to delete.
    }
}
```



```
        """);

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String queueName = args[0];
    SqsClient sqs = SqsClient.builder()
        .region(Region.US_WEST_2)
        .build();

    deleteSQSQueue(sqs, queueName);
    sqs.close();
}

public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
    try {
        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
        DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
            .queueUrl(queueUrl)
            .build();

        sqsClient.deleteQueue(deleteQueueRequest);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [DeleteQueue](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Abrufen der URL einer Warteschlange

Das folgende Codebeispiel zeigt, wie Sie die URL einer Amazon SQS-Warteschlange abrufen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.


```
GetQueueUrlResponse getQueueUrlResponse = sqsClient
    .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
    return getQueueUrlResponse.queueUrl();
```

- Weitere API-Informationen finden Sie unter [GetQueueUrl](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Warteschlangen

Das folgende Codebeispiel zeigt, wie Sie Amazon SQS-Warteschlangen auflisten.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
String prefix = "que";

try {
    ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
    ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
    for (String url : listQueuesResponse.queueUrls()) {
        System.out.println(url);
    }
}
```

```
    } catch (SqsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Weitere API-Informationen finden Sie unter [ListQueues](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Empfangen von Nachrichten aus einer Warteschlange

Das folgende Codebeispiel zeigt, wie Nachrichten aus einer Amazon SQS-Warteschlange empfangen werden.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
    try {  
        ReceiveMessageRequest receiveMessageRequest =  
ReceiveMessageRequest.builder()  
            .queueUrl(queueUrl)  
            .numberOfMessages(5)  
            .build();  
        return sqsClient.receiveMessage(receiveMessageRequest).messages();  
    } catch (SqsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return null;  
}
```

- Weitere API-Informationen finden Sie unter [ReceiveMessage](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Senden eines Stapels von Nachrichten an eine Warteschlange

Das folgende Codebeispiel zeigt, wie Sie einen Stapel von Nachrichten an eine Amazon SQS-Warteschlange senden.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
    .queueUrl(queueUrl)

    .entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from
msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)
        .build())
    .build();
sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

- Weitere API-Informationen finden Sie unter [SendMessageBatch](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Senden einer Nachricht an eine Warteschlange

Das folgende Codebeispiel zeigt, wie Sie eine Nachricht an eine Amazon SQS-Warteschlange senden.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessages {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <queueName> <message>

                Where:
                    queueName - The name of the queue.
                    message - The message to send.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String queueName = args[0];
        String message = args[1];
```

```
SqsClient sqsClient = SqsClient.builder()
    .region(Region.US_WEST_2)
    .build();
sendMessage(sqsClient, queueName, message);
sqsClient.close();
}

public static void sendMessage(SqsClient sqsClient, String queueName, String
message) {
    try {
        CreateQueueRequest request = CreateQueueRequest.builder()
            .queueName(queueName)
            .build();
        sqsClient.createQueue(request);

        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
        SendMessageRequest sendMsgRequest = SendMessageRequest.builder()
            .queueUrl(queueUrl)
            .messageBody(message)
            .delaySeconds(5)
            .build();

        sqsClient.sendMessage(sendMsgRequest);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [SendMessage](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Szenarien

### Erstellen und veröffentlichen zu einem FIFO-Thema

Die folgenden Code-Beispiele zeigen, wie man ein Amazon-SNS-Thema erstellt.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Dieses Beispiel

- erstellt ein Amazon-SNS-FIFO-Thema, zwei Amazon SQS-FIFO-Warteschlangen und eine Standard-Warteschlange.
- abonniert die Warteschlangen für das Thema und veröffentlicht eine Nachricht zu dem Thema.

Der [Test](#) überprüft den Eingang der Nachricht in jeder Warteschlange. Das [vollständige Beispiel](#) zeigt auch das Hinzufügen von Zugriffsrichtlinien und löscht die Ressourcen am Ende.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
<analyticsQueueName>\n\n" +
            "Where:\n" +
            "    fifoTopicName - The name of the FIFO topic that you want to
create. \n\n" +
            "    wholesaleQueueARN - The name of a SQS FIFO queue that will be
created for the wholesale consumer. \n\n"
            +
            "    retailQueueARN - The name of a SQS FIFO queue that will created
for the retail consumer. \n\n" +
            "    analyticsQueueARN - The name of a SQS standard queue that will
be created for the analytics consumer. \n\n";
```

```
    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    final String fifoTopicName = args[0];
    final String wholeSaleQueueName = args[1];
    final String retailQueueName = args[2];
    final String analyticsQueueName = args[3];

    // For convenience, the QueueData class holds metadata about a queue: ARN,
URL,
    // name and type.
    List<QueueData> queues = List.of(
        new QueueData(wholeSaleQueueName, QueueType.FIFO),
        new QueueData(retailQueueName, QueueType.FIFO),
        new QueueData(analyticsQueueName, QueueType.Standard));

    // Create queues.
    createQueues(queues);

    // Create a topic.
    String topicARN = createFIFOTopic(fifoTopicName);

    // Subscribe each queue to the topic.
    subscribeQueues(queues, topicARN);

    // Allow the newly created topic to send messages to the queues.
    addAccessPolicyToQueuesFINAL(queues, topicARN);

    // Publish a sample price update message with payload.
    publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

    // Clean up resources.
    deleteSubscriptions(queues);
    deleteQueues(queues);
    deleteTopic(topicARN);
}

public static String createFIFOTopic(String topicName) {
    try {
        // Create a FIFO topic by using the SNS service client.
        Map<String, String> topicAttributes = Map.of(
```



```
        "FifoTopic", "true",
        "ContentBasedDeduplication", "false");

    CreateTopicRequest topicRequest = CreateTopicRequest.builder()
        .name(topicName)
        .attributes(topicAttributes)
        .build();

    CreateTopicResponse response = snsClient.createTopic(topicRequest);
    String topicArn = response.topicArn();
    System.out.println("The topic ARN is" + topicArn);

    return topicArn;

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

public static void subscribeQueues(List<QueueData> queues, String topicARN) {
    queues.forEach(queue -> {
        SubscribeRequest subscribeRequest = SubscribeRequest.builder()
            .topicArn(topicARN)
            .endpoint(queue.queueARN)
            .protocol("sqs")
            .build();

        // Subscribe to the endpoint by using the SNS service client.
        // Only Amazon SQS queues can receive notifications from an Amazon SNS
        FIFO
        // topic.
        SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
        System.out.println("The queue [" + queue.queueARN + "] subscribed to the
topic [" + topicARN + "]);
        queue.subscriptionARN = subscribeResponse.subscriptionArn();
    });
}

public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {
```

```
try {
    // Create and publish a message that updates the wholesale price.
    String subject = "Price Update";
    String dedupId = UUID.randomUUID().toString();
    String attributeName = "business";
    String attributeValue = "wholesale";

    MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
        .dataType("String")
        .stringValue(attributeValue)
        .build();

    Map<String, MessageAttributeValue> attributes = new HashMap<>();
    attributes.put(attributeName, msgAttValue);
    PublishRequest pubRequest = PublishRequest.builder()
        .topicArn(topicArn)
        .subject(subject)
        .message(payload)
        .messageGroupId(groupId)
        .messageDeduplicationId(dedupId)
        .messageAttributes(attributes)
        .build();

    final PublishResponse response = snsClient.publish(pubRequest);
    System.out.println(response.messageId());
    System.out.println(response.sequenceNumber());
    System.out.println("Message was published to " + topicArn);

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [CreateTopic](#)
  - [Veröffentlichen](#)
  - [Abonnieren](#)

## Serverless-Beispiele

### Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

### Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage;

public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
            context.getLogger().log("An error occurred");
        }
    }
}
```

```
        throw e;
    }
}
}
```

## Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine partielle Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

## Melden von Fehlern bei SQS-Batchelementen mit Lambda unter Verwendung von Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
    SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {

        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
        ArrayList<SQSBatchResponse.BatchItemFailure>();
        String messageId = "";
        for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
            try {
```

```
        //process your message
        messageId = message.getMessageId();
    } catch (Exception e) {
        //Add failed message identifier to the batchItemFailures list
        batchItemFailures.add(new
SQSBatchResponse.BatchItemFailure(messageId));
    }
}
return new SQSBatchResponse(batchItemFailures);
}
}
```

## Step Functions-Beispiele mit SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Step Functions verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

### Erste Schritte

#### Hallo Step Functions

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Step Functions.

#### SDK für Java 2.x

##### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

## Java-Version von Hello.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListStateMachinesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.StateMachineListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListStateMachines {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
            .region(region)
            .build();

        listMachines(sfnClient);
        sfnClient.close();
    }

    public static void listMachines(SfnClient sfnClient) {
        try {
            ListStateMachinesResponse response = sfnClient.listStateMachines();
            List<StateMachineListItem> machines = response.stateMachines();
            for (StateMachineListItem machine : machines) {
                System.out.println("The name of the state machine is: " +
machine.name());
                System.out.println("The ARN value is : " +
machine.stateMachineArn());
            }
        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
}
```

- Weitere API-Informationen finden Sie unter [ListStateMachines](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Themen

- [Aktionen](#)
- [Szenarien](#)

## Aktionen

### Erstellen eines Zustandsautomaten

Das folgende Codebeispiel zeigt, wie Sie einen Step-Functions-Zustandsautomaten erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createMachine(SfnClient sfnClient, String roleARN, String
stateMachineName, String json) {
    try {
        CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
            .definition(json)
            .name(stateMachineName)
            .roleArn(roleARN)
            .type(StateMachineType.STANDARD)
            .build();

        CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
        return response.stateMachineArn();
    }
}
```

```
    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Weitere API-Informationen finden Sie unter [CreateStateMachine](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erstellen einer Aktivität

Das folgende Codebeispiel zeigt, wie Sie eine Step-Functions-Aktivität erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createActivity(SfnClient sfnClient, String activityName) {
    try {
        CreateActivityRequest activityRequest = CreateActivityRequest.builder()
            .name(activityName)
            .build();

        CreateActivityResponse response =
sfnClient.createActivity(activityRequest);
        return response.activityArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```



- Weitere API-Informationen finden Sie unter [CreateActivity](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Löschen eines Zustandsautomaten

Das folgende Codebeispiel zeigt, wie Sie einen Step-Functions-Zustandsautomaten löschen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
    try {
        DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        sfnClient.deleteStateMachine(deleteStateMachineRequest);
        DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        while (true) {
            DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
            System.out.println("The state machine is not deleted yet. The status
is " + response.status());
            Thread.sleep(3000);
        }

    } catch (SfnException | InterruptedException e) {
        System.err.println(e.getMessage());
    }
    System.out.println(stateMachineArn + " was successfully deleted.");
}
```

- Weitere API-Informationen finden Sie unter [DeleteStateMachine](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Löschen einer Aktivität

Das folgende Codebeispiel zeigt, wie Sie eine Step-Functions-Aktivität löschen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void deleteActivity(SfnClient sfnClient, String actArn) {
    try {
        DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()
            .activityArn(actArn)
            .build();

        sfnClient.deleteActivity(activityRequest);
        System.out.println("You have deleted " + actArn);

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DeleteActivity](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Beschreiben eines Zustandsautomaten

Das folgende Codebeispiel zeigt, wie ein Step-Functions-Zustandsautomat beschrieben wird.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void describeStateMachine(SfnClient sfnClient, String
stateMachineArn) {
    try {
        DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
        System.out.println("The name of the State machine is " +
response.name());
        System.out.println("The status of the State machine is " +
response.status());
        System.out.println("The ARN value of the State machine is " +
response.stateMachineArn());
        System.out.println("The role ARN value is " + response.roleArn());

    } catch (SfnException e) {
        System.err.println(e.getMessage());
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeStateMachine](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Beschreiben einer Ausführung eines Zustandsautomaten

Das folgende Codebeispiel zeigt, wie Sie eine Ausführung eines Step Functions-Zustandsautomaten beschreiben.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void describeExe(SfnClient sfnClient, String executionArn) {
    try {
        DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
            .executionArn(executionArn)
            .build();

        String status = "";
        boolean hasSucceeded = false;
        while (!hasSucceeded) {
            DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
            status = response.statusAsString();
            if (status.compareTo("RUNNING") == 0) {
                System.out.println("The state machine is still running, let's
wait for it to finish.");
                Thread.sleep(2000);
            } else if (status.compareTo("SUCCEEDED") == 0) {
                System.out.println("The Step Function workflow has succeeded");
                hasSucceeded = true;
            } else {
                System.out.println("The Status is neither running or
succeeded");
            }
        }
        System.out.println("The Status is " + status);

    } catch (SfnException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeExecution](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Abrufen von Aufgabendaten für eine Aktivität

Das folgende Codebeispiel zeigt, wie Sie Aufgabendaten für eine Step-Functions-Aktivität abrufen.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static List<String> getActivityTask(SfnClient sfnClient, String actArn) {
    List<String> myList = new ArrayList<>();
    GetActivityTaskRequest getActivityTaskRequest =
    GetActivityTaskRequest.builder()
        .activityArn(actArn)
        .build();

    GetActivityTaskResponse response =
    sfnClient.getActivityTask(getActivityTaskRequest);
    myList.add(response.taskToken());
    myList.add(response.input());
    return myList;
}

/// <summary>
/// Stop execution of a Step Functions workflow.
/// </summary>
/// <param name="executionArn">The Amazon Resource Name (ARN) of
/// the Step Functions execution to stop.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> StopExecution(string executionArn)
{
    var response =
        await _amazonStepFunctions.StopExecutionAsync(new StopExecutionRequest
    { ExecutionArn = executionArn });
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- Weitere API-Informationen finden Sie unter [GetActivityTask](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Aktivitäten

Das folgende Codebeispiel zeigt, wie Step-Functions-Aktivitäten aufgelistet werden.

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListActivitiesRequest;
import software.amazon.awssdk.services.sfn.model.ListActivitiesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.ActivityListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListActivities {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
            .region(region)
            .build();

        listAllActivites(sfnClient);
    }
}
```

```
        sfnClient.close();
    }

    public static void listAllActivites(SfnClient sfnClient) {
        try {
            ListActivitiesRequest activitiesRequest =
ListActivitiesRequest.builder()
                .maxResults(10)
                .build();

            ListActivitiesResponse response =
sfnClient.listActivities(activitiesRequest);
            List<ActivityListItem> items = response.activities();
            for (ActivityListItem item : items) {
                System.out.println("The activity ARN is " + item.activityArn());
                System.out.println("The activity name is " + item.name());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [ListActivities](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Zustandsautomatenausführungen

Das folgende Codebeispiel zeigt, wie Sie Step-Functions-Zustandsautomatenausführungen auflisten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void getExeHistory(SfnClient sfnClient, String exeARN) {
```

```
try {
    GetExecutionHistoryRequest historyRequest =
    GetExecutionHistoryRequest.builder()
        .executionArn(exeARN)
        .maxResults(10)
        .build();

    GetExecutionHistoryResponse historyResponse =
    sfnClient.getExecutionHistory(historyRequest);
    List<HistoryEvent> events = historyResponse.events();
    for (HistoryEvent event : events) {
        System.out.println("The event type is " + event.type().toString());
    }

} catch (SfnException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [ListExecutions](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Auflisten von Zustandsautomaten

Das folgende Codebeispiel zeigt, wie Step-Functions-Zustandsautomaten aufgelistet werden.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListStateMachinesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.StateMachineListItem;
```



```
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListStateMachines {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
            .region(region)
            .build();

        listMachines(sfnClient);
        sfnClient.close();
    }

    public static void listMachines(SfnClient sfnClient) {
        try {
            ListStateMachinesResponse response = sfnClient.listStateMachines();
            List<StateMachineListItem> machines = response.stateMachines();
            for (StateMachineListItem machine : machines) {
                System.out.println("The name of the state machine is: " +
machine.name());
                System.out.println("The ARN value is : " +
machine.stateMachineArn());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [ListStateMachines](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Senden einer Erfolgsantwort an eine Aufgabe

Das folgende Codebeispiel zeigt, wie Sie eine Erfolgsantwort an eine Step-Functions-Aufgabe senden.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void sendTaskSuccess(SfnClient sfnClient, String token, String
json) {
    try {
        SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
            .taskToken(token)
            .output(json)
            .build();

        sfnClient.sendTaskSuccess(successRequest);

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [SendTaskSuccess](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Starten einer Ausführung eines Zustandsautomaten

Das folgende Codebeispiel zeigt, wie Sie eine Ausführung eines Step Functions-Zustandsautomaten starten.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
String jsonEx) {
    UUID uuid = UUID.randomUUID();
    String uuidValue = uuid.toString();
    try {
        StartExecutionRequest executionRequest = StartExecutionRequest.builder()
            .input(jsonEx)
            .stateMachineArn(stateMachineArn)
            .name(uuidValue)
            .build();

        StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
        return response.executionArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Weitere API-Informationen finden Sie unter [StartExecution](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Szenarien

### Erste Schritte mit Zustandsautomaten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine -Aktivität.

- Erstellen Sie einen Zustandsautomaten aus einer Amazon States Language-Definition, die die zuvor erstellte Aktivität als Schritt enthält.
- Führen Sie den Zustandsautomaten aus und reagieren Sie auf die Aktivität mit Benutzereingabe.
- Rufen Sie nach Abschluss der Ausführung den endgültigen Status und die Ausgabe ab und bereinigen Sie dann die Ressourcen.

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * You can obtain the JSON file to create a state machine in the following
 * GitHub location.
 *
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/sample_files
 *
 * To run this code example, place the chat_sfn_state_machine.json file into
 * your project's resources folder.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an activity.
 * 2. Creates a state machine.
 * 3. Describes the state machine.
 * 4. Starts execution of the state machine and interacts with it.
 * 5. Describes the execution.
 * 6. Delete the activity.
 * 7. Deletes the state machine.
 */
public class StepFunctionsScenario {
```

```

public static final String DASHES = new String(new char[80]).replace("\0", "-");

public static void main(String[] args) throws Exception {
    final String usage = ""

        Usage:
            <roleARN> <activityName> <stateMachineName>

        Where:
            roleName - The name of the IAM role to create for this state
machine.

            activityName - The name of an activity to create.
            stateMachineName - The name of the state machine to create.
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String roleName = args[0];
    String activityName = args[1];
    String stateMachineName = args[2];
    String polJSON = "{\n" +
        "    \"Version\": \"2012-10-17\",\n" +
        "    \"Statement\": [\n" +
        "        {\n" +
        "            \"Sid\": \"\",\n" +
        "            \"Effect\": \"Allow\",\n" +
        "            \"Principal\": {\n" +
        "                \"Service\": \"states.amazonaws.com\"\n" +
        "            },\n" +
        "            \"Action\": \"sts:AssumeRole\"\n" +
        "        }\n" +
        "    ]\n" +
        "}";

    Scanner sc = new Scanner(System.in);
    boolean action = false;

    Region region = Region.US_EAST_1;
    SfnClient sfnClient = SfnClient.builder()
        .region(region)
        .build();

```

```
Region regionGl = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(regionGl)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the AWS Step Functions example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create an activity.");
String activityArn = createActivity(sfnClient, activityName);
System.out.println("The ARN of the activity is " + activityArn);
System.out.println(DASHES);

// Get JSON to use for the state machine and place the activityArn value
into
// it.
InputStream input = StepFunctionsScenario.class.getClassLoader()
    .getResourceAsStream("chat_sfn_state_machine.json");
ObjectMapper mapper = new ObjectMapper();
JsonNode jsonNode = mapper.readValue(input, JsonNode.class);
String jsonString = mapper.writeValueAsString(jsonNode);

// Modify the Resource node.
ObjectMapper objectMapper = new ObjectMapper();
JsonNode root = objectMapper.readTree(jsonString);
((ObjectNode) root.path("States").path("GetInput")).put("Resource",
activityArn);

// Convert the modified Java object back to a JSON string.
String stateDefinition = objectMapper.writeValueAsString(root);
System.out.println(stateDefinition);

System.out.println(DASHES);
System.out.println("2. Create a state machine.");
String roleARN = createIAMRole(iam, roleName, polJSON);
String stateMachineArn = createMachine(sfnClient, roleARN, stateMachineName,
stateDefinition);
System.out.println("The ARN of the state machine is " + stateMachineArn);
System.out.println(DASHES);

System.out.println(DASHES);
```

```
System.out.println("3. Describe the state machine.");
describeStateMachine(sfnClient, stateMachineArn);
System.out.println("What should ChatSFN call you?");
String userName = sc.nextLine();
System.out.println("Hello " + userName);
System.out.println(DASHES);

System.out.println(DASHES);
// The JSON to pass to the StartExecution call.
String executionJson = "{ \"name\" : \"" + userName + "\" }";
System.out.println(executionJson);
System.out.println("4. Start execution of the state machine and interact
with it.");
String runArn = startWorkflow(sfnClient, stateMachineArn, executionJson);
System.out.println("The ARN of the state machine execution is " + runArn);
List<String> myList;
while (!action) {
    myList = getActivityTask(sfnClient, activityArn);
    System.out.println("ChatSFN: " + myList.get(1));
    System.out.println(userName + " please specify a value.");
    String myAction = sc.nextLine();
    if (myAction.compareTo("done") == 0)
        action = true;

    System.out.println("You have selected " + myAction);
    String taskJson = "{ \"action\" : \"" + myAction + "\" }";
    System.out.println(taskJson);
    sendTaskSuccess(sfnClient, myList.get(0), taskJson);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Describe the execution.");
describeExe(sfnClient, runArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Delete the activity.");
deleteActivity(sfnClient, activityArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Delete the state machines.");
deleteMachine(sfnClient, stateMachineArn);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS Step Functions example scenario is complete.");
        System.out.println(DASHES);
    }

    public static String createIAMRole(IamClient iam, String rolename, String
polJSON) {
        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                .roleName(rolename)
                .assumeRolePolicyDocument(polJSON)
                .description("Created using the AWS SDK for Java")
                .build();

            CreateRoleResponse response = iam.createRole(request);
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void describeExe(SfnClient sfnClient, String executionArn) {
        try {
            DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
                .executionArn(executionArn)
                .build();

            String status = "";
            boolean hasSucceeded = false;
            while (!hasSucceeded) {
                DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
                status = response.statusAsString();
                if (status.compareTo("RUNNING") == 0) {
                    System.out.println("The state machine is still running, let's
wait for it to finish.");
                    Thread.sleep(2000);
                } else if (status.compareTo("SUCCEEDED") == 0) {
```



```
        System.out.println("The Step Function workflow has succeeded");
        hasSucceeded = true;
    } else {
        System.out.println("The Status is neither running or
succeeded");
    }
}
System.out.println("The Status is " + status);

} catch (SfnException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void sendTaskSuccess(SfnClient sfnClient, String token, String
json) {
    try {
        SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
            .taskToken(token)
            .output(json)
            .build();

        sfnClient.sendTaskSuccess(successRequest);

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static List<String> getActivityTask(SfnClient sfnClient, String actArn) {
    List<String> myList = new ArrayList<>();
    GetActivityTaskRequest getActivityTaskRequest =
GetActivityTaskRequest.builder()
        .activityArn(actArn)
        .build();

    GetActivityTaskResponse response =
sfnClient.getActivityTask(getActivityTaskRequest);
    myList.add(response.taskToken());
    myList.add(response.input());
    return myList;
}
```

```
public static void deleteActivity(SfnClient sfnClient, String actArn) {
    try {
        DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()
            .activityArn(actArn)
            .build();

        sfnClient.deleteActivity(activityRequest);
        System.out.println("You have deleted " + actArn);

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeStateMachine(SfnClient sfnClient, String
stateMachineArn) {
    try {
        DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
        System.out.println("The name of the State machine is " +
response.name());
        System.out.println("The status of the State machine is " +
response.status());
        System.out.println("The ARN value of the State machine is " +
response.stateMachineArn());
        System.out.println("The role ARN value is " + response.roleArn());

    } catch (SfnException e) {
        System.err.println(e.getMessage());
    }
}

public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
    try {
        DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
```

```
        .build();

        sfnClient.deleteStateMachine(deleteStateMachineRequest);
        DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
        .stateMachineArn(stateMachineArn)
        .build();

        while (true) {
            DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
            System.out.println("The state machine is not deleted yet. The status
is " + response.status());
            Thread.sleep(3000);
        }

    } catch (SfnException | InterruptedException e) {
        System.err.println(e.getMessage());
    }
    System.out.println(stateMachineArn + " was successfully deleted.");
}

public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
String jsonEx) {
    UUID uuid = UUID.randomUUID();
    String uuidValue = uuid.toString();
    try {
        StartExecutionRequest executionRequest = StartExecutionRequest.builder()
            .input(jsonEx)
            .stateMachineArn(stateMachineArn)
            .name(uuidValue)
            .build();

        StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
        return response.executionArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

```
public static String createMachine(SfnClient sfnClient, String roleARN, String
stateMachineName, String json) {
    try {
        CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
            .definition(json)
            .name(stateMachineName)
            .roleArn(roleARN)
            .type(StateMachineType.STANDARD)
            .build();

        CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
        return response.stateMachineArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String createActivity(SfnClient sfnClient, String activityName) {
    try {
        CreateActivityRequest activityRequest = CreateActivityRequest.builder()
            .name(activityName)
            .build();

        CreateActivityResponse response =
sfnClient.createActivity(activityRequest);
        return response.activityArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
- [CreateActivity](#)

- [CreateStateMachine](#)
- [DeleteActivity](#)
- [DeleteStateMachine](#)
- [DescribeExecution](#)
- [DescribeStateMachine](#)
- [GetActivityTask](#)
- [ListActivities](#)
- [ListStateMachines](#)
- [SendTaskSuccess](#)
- [StartExecution](#)
- [StopExecution](#)

## AWS STS -Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit verwenden AWS STS.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)

### Aktionen

#### Übernehmen einer Rolle

Das folgende Codebeispiel zeigt, wie Sie eine Rolle mit übernehmen AWS STS.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.AssumeRoleRequest;
import software.amazon.awssdk.services.sts.model.StsException;
import software.amazon.awssdk.services.sts.model.AssumeRoleResponse;
import software.amazon.awssdk.services.sts.model.Credentials;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * To make this code example work, create a Role that you want to assume.
 * Then define a Trust Relationship in the AWS Console. You can use this as an
 * example:
 *
 * {
 *   "Version": "2012-10-17",
 *   "Statement": [
 *     {
 *       "Effect": "Allow",
 *       "Principal": {
 *         "AWS": "<Specify the ARN of your IAM user you are using in this code
 * example>"
 *       },
 *       "Action": "sts:AssumeRole"
 *     }
 *   ]
 * }
 *
 * For more information, see "Editing the Trust Relationship for an Existing
 * Role" in the AWS Directory Service guide.
 */
```

```
* Also, set up your development environment, including your credentials.
*
* For information, see this documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class AssumeRole {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <roleArn> <roleSessionName>\s

            Where:
                roleArn - The Amazon Resource Name (ARN) of the role to assume
(for example, rn:aws:iam::000008047983:role/s3role).\s
                roleSessionName - An identifier for the assumed role session
(for example, mysession).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleArn = args[0];
        String roleSessionName = args[1];
        Region region = Region.US_EAST_1;
        StsClient stsClient = StsClient.builder()
            .region(region)
            .build();

        assumeGivenRole(stsClient, roleArn, roleSessionName);
        stsClient.close();
    }

    public static void assumeGivenRole(StsClient stsClient, String roleArn, String
roleSessionName) {
        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                .roleArn(roleArn)
                .roleSessionName(roleSessionName)
                .build();
```

```
AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
Credentials myCreds = roleResponse.credentials();

// Display the time when the temp creds expire.
Instant exTime = myCreds.expiration();
String tokenInfo = myCreds.sessionToken();

// Convert the Instant to readable date.
DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
    .withLocale(Locale.US)
    .withZone(ZoneId.systemDefault());

formatter.format(exTime);
System.out.println("The token " + tokenInfo + " expires on " + exTime);

} catch (StsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [AssumeRole](#) in der APIAWS SDK for Java 2.x - Referenz für .

## AWS Support -Beispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit verwenden AWS Support.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.



## Erste Schritte

### Hallo AWS Support

Die folgenden Codebeispiele veranschaulichen, wie Sie mit der Verwendung von AWS Support beginnen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import software.amazon.awssdk.services.support.model.Service;
import software.amazon.awssdk.services.support.model.SupportException;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS
 * Support Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following task:
 *
 * 1. Gets and displays available services.
 *
 *
 */
```

```
* NOTE: To see multiple operations, see SupportScenario.
```

```
*/
```

```
public class HelloSupport {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SupportClient supportClient = SupportClient.builder()
            .region(region)
            .build();

        System.out.println("***** Step 1. Get and display available services.");
        displayServices(supportClient);
    }

    // Return a List that contains a Service name and Category name.
    public static void displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
                .language("en")
                .build();

            DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service : services) {
                if (index == 11)
                    break;

                System.out.println("The Service name is: " + service.name());

                // Display the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat : categories) {
                    System.out.println("The category name is: " + cat.name());
                }
                index++;
            }

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
        }
    }
}
```

```
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [DescribeServices](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Themen

- [Aktionen](#)
- [Szenarien](#)

## Aktionen

### Hinzufügen einer Mitteilung zu einem Fall

Das folgende Codebeispiel zeigt, wie Sie einem Support-Fall eine - AWS Support Kommunikation mit einem -Anhang hinzufügen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
    try {
        AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
            .caseId(caseId)
            .attachmentSetId(attachmentSetId)
            .communicationBody("Please refer to attachment for details.")
            .build();

        AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
    }
}
```

```
        if (response.result())
            System.out.println("You have successfully added a communication to
an AWS Support case");
        else
            System.out.println("There was an error adding the communication to
an AWS Support case");

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [AddCommunicationToCase](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Hinzufügen eines Anhangs zu einem Set

Das folgende Codebeispiel zeigt, wie Sie einem Anhangssatz einen AWS Support Anhang hinzufügen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
    try {
        File myFile = new File(fileAttachment);
        InputStream sourceStream = new FileInputStream(myFile);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        Attachment attachment = Attachment.builder()
            .fileName(myFile.getName())
            .data(sourceBytes)
            .build();
```

```
        AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
        .attachments(attachment)
        .build();

        AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
        return response.attachmentSetId();

    } catch (SupportException | FileNotFoundException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

- Weitere API-Informationen finden Sie unter [AddAttachmentsToSet](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erstellen eines Falls

Das folgende Codebeispiel zeigt, wie Sie einen neuen AWS Support Fall erstellen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String createSupportCase(SupportClient supportClient, List<String>
sevCatList, String sevLevel) {
    try {
        String serviceCode = sevCatList.get(0);
        String caseCat = sevCatList.get(1);
        CreateCaseRequest caseRequest = CreateCaseRequest.builder()
            .categoryCode(caseCat.toLowerCase())
            .serviceCode(serviceCode.toLowerCase())
```

```
        .severityCode(sevLevel.toLowerCase())
        .communicationBody("Test issue with " +
serviceCode.toLowerCase())
        .subject("Test case, please ignore")
        .language("en")
        .issueType("technical")
        .build();

    CreateCaseResponse response = supportClient.createCase(caseRequest);
    return response.caseId();

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return "";
}
```

- Weitere API-Informationen finden Sie unter [CreateCase](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beschreiben eines Anhangs

Das folgende Codebeispiel zeigt, wie Sie einen Anhang für einen AWS Support -Fall beschreiben.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void describeAttachment(SupportClient supportClient, String
attachId) {
    try {
        DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
            .attachmentId(attachId)
            .build();
```

```
        DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
        System.out.println("The name of the file is " +
response.attachment().fileName());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeAttachment](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Beschreiben von Fällen

Das folgende Codebeispiel zeigt, wie Sie - AWS Support Fälle beschreiben.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void getOpenCase(SupportClient supportClient) {
    try {
        // Specify the start and end time.
        Instant now = Instant.now();
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);

        DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
            .maxResults(20)
            .afterTime(yesterday.toString())
            .beforeTime(now.toString())
            .build();
```

```
DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
List<CaseDetails> cases = response.cases();
for (CaseDetails sinCase : cases) {
    System.out.println("The case status is " + sinCase.status());
    System.out.println("The case Id is " + sinCase.caseId());
    System.out.println("The case subject is " + sinCase.subject());
}

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- Weitere API-Informationen finden Sie unter [DescribeCases](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beschreiben von Mitteilungen

Das folgende Codebeispiel zeigt, wie Sie die AWS Support Kommunikation für einen Fall beschreiben.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String listCommunications(SupportClient supportClient, String
caseId) {
    try {
        String attachId = null;
        DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
            .caseId(caseId)
            .maxResults(10)
            .build();
```



```
DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
List<Communication> communications = response.communications();
for (Communication comm : communications) {
    System.out.println("the body is: " + comm.body());

    // Get the attachment id value.
    List<AttachmentDetails> attachments = comm.attachmentSet();
    for (AttachmentDetails detail : attachments) {
        attachId = detail.attachmentId();
    }
}
return attachId;

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return "";
}
```

- Weitere API-Informationen finden Sie unter [DescribeCommunications](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Beschreiben von Services

Das folgende Codebeispiel zeigt, wie Sie die Liste der - AWS Services beschreiben.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
// Return a List that contains a Service name and Category name.
public static List<String> displayServices(SupportClient supportClient) {
    try {
```

```
DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
    .language("en")
    .build();

DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
String serviceCode = null;
String catName = null;
List<String> sevCatList = new ArrayList<>();
List<Service> services = response.services();

System.out.println("Get the first 10 services");
int index = 1;
for (Service service : services) {
    if (index == 11)
        break;

    System.out.println("The Service name is: " + service.name());
    if (service.name().compareTo("Account") == 0)
        serviceCode = service.code();

    // Get the Categories for this service.
    List<Category> categories = service.categories();
    for (Category cat : categories) {
        System.out.println("The category name is: " + cat.name());
        if (cat.name().compareTo("Security") == 0)
            catName = cat.name();
    }
    index++;
}

// Push the two values to the list.
sevCatList.add(serviceCode);
sevCatList.add(catName);
return sevCatList;

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return null;
}
```

- Weitere API-Informationen finden Sie unter [DescribeServices](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beschreiben der Schweregrade

Das folgende Codebeispiel zeigt, wie Sie AWS Support Schweregrade beschreiben.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static String displaySevLevels(SupportClient supportClient) {
    try {
        DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
            .language("en")
            .build();

        DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
        List<SeverityLevel> severityLevels = response.severityLevels();
        String levelName = null;
        for (SeverityLevel sevLevel : severityLevels) {
            System.out.println("The severity level name is: " +
sevLevel.name());
            if (sevLevel.name().compareTo("High") == 0)
                levelName = sevLevel.name();
        }
        return levelName;

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

```
}
```

- Weitere API-Informationen finden Sie unter [DescribeSeverityLevels](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Lösen eines Falls

Das folgende Codebeispiel zeigt, wie Sie einen - AWS Support Fall lösen.

## SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
    try {
        ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
            .caseId(caseId)
            .build();

        ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
        System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [ResolveCase](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Szenarien

### Erste Schritte mit Fällen

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 Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Führen Sie verschiedene AWS Support Operationen aus.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetResponse;
import software.amazon.awssdk.services.support.model.AddCommunicationToCaseRequest;
import software.amazon.awssdk.services.support.model.AddCommunicationToCaseResponse;
import software.amazon.awssdk.services.support.model.Attachment;
import software.amazon.awssdk.services.support.model.AttachmentDetails;
import software.amazon.awssdk.services.support.model.CaseDetails;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.Communication;
import software.amazon.awssdk.services.support.model.CreateCaseRequest;
import software.amazon.awssdk.services.support.model.CreateCaseResponse;
import software.amazon.awssdk.services.support.model.DescribeAttachmentRequest;
```

```
import software.amazon.awssdk.services.support.model.DescribeAttachmentResponse;
import software.amazon.awssdk.services.support.model.DescribeCasesRequest;
import software.amazon.awssdk.services.support.model.DescribeCasesResponse;
import software.amazon.awssdk.services.support.model.DescribeCommunicationsRequest;
import software.amazon.awssdk.services.support.model.DescribeCommunicationsResponse;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import software.amazon.awssdk.services.support.model.DescribeSeverityLevelsRequest;
import software.amazon.awssdk.services.support.model.DescribeSeverityLevelsResponse;
import software.amazon.awssdk.services.support.model.ResolveCaseRequest;
import software.amazon.awssdk.services.support.model.ResolveCaseResponse;
import software.amazon.awssdk.services.support.model.Service;
import software.amazon.awssdk.services.support.model.SeverityLevel;
import software.amazon.awssdk.services.support.model.SupportException;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetRequest;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.time.Instant;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS
 * Support Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets and displays available services.
 * 2. Gets and displays severity levels.
 * 3. Creates a support case by using the selected service, category, and
 * severity level.
 * 4. Gets a list of open cases for the current day.
```

```
* 5. Creates an attachment set with a generated file.
* 6. Adds a communication with the attachment to the support case.
* 7. Lists the communications of the support case.
* 8. Describes the attachment set included with the communication.
* 9. Resolves the support case.
* 10. Gets a list of resolved cases for the current day.
*/
public class SupportScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <fileAttachment>Where:
            fileAttachment - The file can be a simple saved .txt file to use
as an email attachment.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String fileAttachment = args[0];
        Region region = Region.US_WEST_2;
        SupportClient supportClient = SupportClient.builder()
            .region(region)
            .build();

        System.out.println(DASHES);
        System.out.println("***** Welcome to the AWS Support case example
scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Get and display available services.");
        List<String> sevCatList = displayServices(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Get and display Support severity levels.");
        String sevLevel = displaySevLevels(supportClient);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a support case using the selected service,
category, and severity level.");
        String caseId = createSupportCase(supportClient, sevCatList, sevLevel);
        if (caseId.compareTo("") == 0) {
            System.out.println("A support case was not successfully created!");
            System.exit(1);
        } else
            System.out.println("Support case " + caseId + " was successfully
created!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get open support cases.");
        getOpenCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create an attachment set with a generated file to add
to the case.");
        String attachmentSetId = addAttachment(supportClient, fileAttachment);
        System.out.println("The Attachment Set id value is" + attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add communication with the attachment to the support
case.");
        addAttachSupportCase(supportClient, caseId, attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. List the communications of the support case.");
        String attachId = listCommunications(supportClient, caseId);
        System.out.println("The Attachment id value is" + attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Describe the attachment set included with the
communication.");
        describeAttachment(supportClient, attachId);
        System.out.println(DASHES);
```



```
        System.out.println(DASHES);
        System.out.println("9. Resolve the support case.");
        resolveSupportCase(supportClient, caseId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of resolved cases for the current day.");
        getResolvedCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("***** This Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static void getResolvedCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
            java.time.LocalDate.now();
            Instant yesterday = now.minus(1, ChronoUnit.DAYS);

            DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
                .maxResults(30)
                .afterTime(yesterday.toString())
                .beforeTime(now.toString())
                .includeResolvedCases(true)
                .build();

            DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
            List<CaseDetails> cases = response.cases();
            for (CaseDetails sinCase : cases) {
                if (sinCase.status().compareTo("resolved") == 0)
                    System.out.println("The case status is " + sinCase.status());
            }

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

```
public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
    try {
        ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
            .caseId(caseId)
            .build();

        ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
        System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeAttachment(SupportClient supportClient, String
attachId) {
    try {
        DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
            .attachmentId(attachId)
            .build();

        DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
        System.out.println("The name of the file is " +
response.attachment().fileName());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static String listCommunications(SupportClient supportClient, String
caseId) {
    try {
        String attachId = null;
        DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
            .caseId(caseId)
            .maxResults(10)
```

```
        .build();

        DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
        List<Communication> communications = response.communications();
        for (Communication comm : communications) {
            System.out.println("the body is: " + comm.body());

            // Get the attachment id value.
            List<AttachmentDetails> attachments = comm.attachmentSet();
            for (AttachmentDetails detail : attachments) {
                attachId = detail.attachmentId();
            }
        }
        return attachId;

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
    try {
        AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
            .caseId(caseId)
            .attachmentSetId(attachmentSetId)
            .communicationBody("Please refer to attachment for details.")
            .build();

        AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
        if (response.result())
            System.out.println("You have successfully added a communication to
an AWS Support case");
        else
            System.out.println("There was an error adding the communication to
an AWS Support case");

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
    }
}
```

```
        System.exit(1);
    }
}

public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
    try {
        File myFile = new File(fileAttachment);
        InputStream sourceStream = new FileInputStream(myFile);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        Attachment attachment = Attachment.builder()
            .fileName(myFile.getName())
            .data(sourceBytes)
            .build();

        AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
            .attachments(attachment)
            .build();

        AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
        return response.attachmentSetId();

    } catch (SupportException | FileNotFoundException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

public static void getOpenCase(SupportClient supportClient) {
    try {
        // Specify the start and end time.
        Instant now = Instant.now();
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);

        DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
            .maxResults(20)
            .afterTime(yesterday.toString())
            .beforeTime(now.toString())
```

```
        .build();

        DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
        List<CaseDetails> cases = response.cases();
        for (CaseDetails sinCase : cases) {
            System.out.println("The case status is " + sinCase.status());
            System.out.println("The case Id is " + sinCase.caseId());
            System.out.println("The case subject is " + sinCase.subject());
        }

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static String createSupportCase(SupportClient supportClient, List<String>
sevCatList, String sevLevel) {
    try {
        String serviceCode = sevCatList.get(0);
        String caseCat = sevCatList.get(1);
        CreateCaseRequest caseRequest = CreateCaseRequest.builder()
            .categoryCode(caseCat.toLowerCase())
            .serviceCode(serviceCode.toLowerCase())
            .severityCode(sevLevel.toLowerCase())
            .communicationBody("Test issue with " +
serviceCode.toLowerCase())
            .subject("Test case, please ignore")
            .language("en")
            .issueType("technical")
            .build();

        CreateCaseResponse response = supportClient.createCase(caseRequest);
        return response.caseId();

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

public static String displaySevLevels(SupportClient supportClient) {
```

```
        try {
            DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
                .language("en")
                .build();

            DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
            List<SeverityLevel> severityLevels = response.severityLevels();
            String levelName = null;
            for (SeverityLevel sevLevel : severityLevels) {
                System.out.println("The severity level name is: " +
sevLevel.name());
                if (sevLevel.name().compareTo("High") == 0)
                    levelName = sevLevel.name();
            }
            return levelName;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    // Return a List that contains a Service name and Category name.
    public static List<String> displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
                .language("en")
                .build();

            DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
            String serviceCode = null;
            String catName = null;
            List<String> sevCatList = new ArrayList<>();
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service : services) {
                if (index == 11)
```

```
        break;

        System.out.println("The Service name is: " + service.name());
        if (service.name().compareTo("Account") == 0)
            serviceCode = service.code();

        // Get the Categories for this service.
        List<Category> categories = service.categories();
        for (Category cat : categories) {
            System.out.println("The category name is: " + cat.name());
            if (cat.name().compareTo("Security") == 0)
                catName = cat.name();
        }
        index++;
    }

    // Push the two values to the list.
    sevCatList.add(serviceCode);
    sevCatList.add(catName);
    return sevCatList;

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return null;
}
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [AddAttachmentsToSet](#)
  - [AddCommunicationToCase](#)
  - [CreateCase](#)
  - [DescribeAttachment](#)
  - [DescribeCases](#)
  - [DescribeCommunications](#)
  - [DescribeServices](#)
  - [DescribeSeverityLevels](#)
  - [ResolveCase](#)

## Systems Manager-Beispiele mit SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Systems Manager verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)

### Aktionen

Hinzufügen eines Parameters

Das folgende Codebeispiel zeigt, wie Sie einen Systems Manager-Parameter hinzufügen.

SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.ParameterType;
import software.amazon.awssdk.services.ssm.model.PutParameterRequest;
import software.amazon.awssdk.services.ssm.model.SsmException;

public class PutParameter {

    public static void main(String[] args) {
```



```
final String usage = ""

    Usage:
        <paraName>

    Where:
        paraName - The name of the parameter.
        paraValue - The value of the parameter.
    """;

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String paraName = args[0];
String paraValue = args[1];
Region region = Region.US_EAST_1;
SsmClient ssmClient = SsmClient.builder()
    .region(region)
    .build();

putParaValue(ssmClient, paraName, paraValue);
ssmClient.close();
}

public static void putParaValue(SsmClient ssmClient, String paraName, String
value) {
    try {
        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(paraName)
            .type(ParameterType.STRING)
            .value(value)
            .build();

        ssmClient.putParameter(parameterRequest);
        System.out.println("The parameter was successfully added.");

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [PutParameter](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Erstellen eines neuen OpsItem

Das folgende Codebeispiel zeigt, wie Sie ein neues erstellen OpsItem.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.CreateOpsItemRequest;
import software.amazon.awssdk.services.ssm.model.CreateOpsItemResponse;
import software.amazon.awssdk.services.ssm.model.SsmException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateOpsItem {
    public static void main(String[] args) {

        final String USAGE = ""

            Usage:
                <title> <source> <category> <severity>

            Where:
                title - The OpsItem title.
```

```
        source - The origin of the OpsItem, such as Amazon EC2 or AWS
Systems Manager.
        category - A category to assign to an OpsItem.
        severity - A severity to assign to an OpsItem.
        """;

    if (args.length != 4) {
        System.out.println(USAGE);
        System.exit(1);
    }

    String title = args[0];
    String source = args[1];
    String category = args[2];
    String severity = args[3];

    Region region = Region.US_EAST_1;
    SsmClient ssmClient = SsmClient.builder()
        .region(region)
        .build();

    System.out
        .println("The Id of the OpsItem is " + createNewOpsItem(ssmClient,
title, source, category, severity));
    ssmClient.close();
}

public static String createNewOpsItem(SsmClient ssmClient,
    String title,
    String source,
    String category,
    String severity) {

    try {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
            .description("Created by the SSM Java API")
            .title(title)
            .source(source)
            .category(category)
            .severity(severity)
            .build();

        CreateOpsItemResponse itemResponse =
ssmClient.createOpsItem(opsItemRequest);
```

```
        return itemResponse.opsItemId();

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Weitere API-Informationen finden Sie unter [CreateOpsItem](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Beschreiben eines OpsItem

Das folgende Codebeispiel zeigt, wie Sie eine beschreiben OpsItem.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.DescribeOpsItemsRequest;
import software.amazon.awssdk.services.ssm.model.DescribeOpsItemsResponse;
import software.amazon.awssdk.services.ssm.model.OpsItemSummary;
import software.amazon.awssdk.services.ssm.model.SsmException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
*/
public class DescribeOpsItems {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();

        describeItems(ssmClient);
        ssmClient.close();
    }

    public static void describeItems(SsmClient ssmClient) {
        try {
            DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
                .maxResults(10)
                .build();

            DescribeOpsItemsResponse itemsResponse =
                ssmClient.describeOpsItems(itemsRequest);
            List<OpsItemSummary> items = itemsResponse.opsItemSummaries();
            for (OpsItemSummary item : items) {
                System.out.println("The item title is " + item.title());
            }

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeOpsItems](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Abrufen von Parameterinformationen

Das folgende Codebeispiel zeigt, wie Sie Systems Manager-Parameterinformationen abrufen.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.GetParameterRequest;
import software.amazon.awssdk.services.ssm.model.GetParameterResponse;
import software.amazon.awssdk.services.ssm.model.SsmException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetParameter {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <paraName>

            Where:
                paraName - The name of the parameter.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String paraName = args[0];
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
```

```
        .build();

        getParaValue(ssmClient, paraName);
        ssmClient.close();
    }

    public static void getParaValue(SsmClient ssmClient, String paraName) {
        try {
            GetParameterRequest parameterRequest = GetParameterRequest.builder()
                .name(paraName)
                .build();

            GetParameterResponse parameterResponse =
                ssmClient.getParameter(parameterRequest);
            System.out.println("The parameter value is " +
                parameterResponse.parameter().value());


        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter [DescribeParameters](#) in der APIAWS SDK for Java 2.x -Referenz für .

Aktualisiert ein OpsItem

Das folgende Codebeispiel zeigt, wie Sie eine aktualisieren OpsItem.

SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
```

```
import software.amazon.awssdk.services.ssm.model.SsmException;
import software.amazon.awssdk.services.ssm.model.UpdateOpsItemRequest;
import software.amazon.awssdk.services.ssm.model.OpsItemStatus;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ResolveOpsItem {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <opsID>

            Where:
                opsID - The Ops item ID value.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String opsID = args[0];
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();
        setOpsItemStatus(ssmClient, opsID);
    }

    public static void setOpsItemStatus(SsmClient ssmClient, String opsID) {
        try {
            UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
                .opsItemId(opsID)
                .status(OpsItemStatus.RESOLVED)
                .build();

            ssmClient.updateOpsItem(opsItemRequest);
        }
    }
}
```



```
        } catch (SsmException e) {  
            System.err.println(e.getMessage());  
            System.exit(1);  
        }  
    }  
}
```

- Weitere API-Informationen finden Sie unter [UpdateOpsItem](#) in der APIAWS SDK for Java 2.x - Referenz für .

## Amazon-Textract-Beispiele unter Verwendung von SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon Textract verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)

### Aktionen

#### Analysieren eines Dokuments

Das folgende Codebeispiel zeigt, wie Sie ein Dokument mit Amazon Textract analysieren.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.AnalyzeDocumentRequest;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.FeatureType;
import software.amazon.awssdk.services.textract.model.AnalyzeDocumentResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.TextractException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AnalyzeDocument {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <sourceDoc>\s

                Where:
```

```
        sourceDoc - The path where the document is located (must be an
image, for example, C:/AWS/book.png).\s
        """";

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String sourceDoc = args[0];
    Region region = Region.US_EAST_2;
    TextractClient textractClient = TextractClient.builder()
        .region(region)
        .build();

    analyzeDoc(textractClient, sourceDoc);
    textractClient.close();
}

public static void analyzeDoc(TextractClient textractClient, String sourceDoc) {
    try {
        InputStream sourceStream = new FileInputStream(new File(sourceDoc));
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        // Get the input Document object as bytes
        Document myDoc = Document.builder()
            .bytes(sourceBytes)
            .build();

        List<FeatureType> featureTypes = new ArrayList<FeatureType>();
        featureTypes.add(FeatureType.FORMS);
        featureTypes.add(FeatureType.TABLES);

        AnalyzeDocumentRequest analyzeDocumentRequest =
    AnalyzeDocumentRequest.builder()
        .featureTypes(featureTypes)
        .document(myDoc)
        .build();

        AnalyzeDocumentResponse analyzeDocument =
    textractClient.analyzeDocument(analyzeDocumentRequest);
        List<Block> docInfo = analyzeDocument.blocks();
        Iterator<Block> blockIterator = docInfo.iterator();
    }
```

```
        while (blockIterator.hasNext()) {
            Block block = blockIterator.next();
            System.out.println("The block type is " +
block.blockType().toString());
        }

    } catch (TextractException | FileNotFoundException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Weitere API-Informationen finden Sie unter [AnalyzeDocument](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Erkennen von Text in einem Dokument

Das folgende Codebeispiel zeigt, wie Sie mit Amazon Textract Text in einem Dokument erkennen.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Erkennen Sie Text aus einem Eingabedokument.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
import software.amazon.awssdk.services.textract.model.TextractException;
import java.io.File;
```

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectDocumentText {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <sourceDoc>\s

                Where:
                sourceDoc - The path where the document is located (must be an
image, for example, C:/AWS/book.png).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceDoc = args[0];
        Region region = Region.US_EAST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        detectDocText(textractClient, sourceDoc);
        textractClient.close();
    }

    public static void detectDocText(TextractClient textractClient, String
sourceDoc) {
        try {
            InputStream sourceStream = new FileInputStream(new File(sourceDoc));
```

```
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        // Get the input Document object as bytes.
        Document myDoc = Document.builder()
            .bytes(sourceBytes)
            .build();

        DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
            .document(myDoc)
            .build();

        // Invoke the Detect operation.
        DetectDocumentTextResponse textResponse =
textextractClient.detectDocumentText(detectDocumentTextRequest);
        List<Block> docInfo = textResponse.blocks();
        for (Block block : docInfo) {
            System.out.println("The block type is " +
block.blockType().toString());
        }

        DocumentMetadata documentMetadata = textResponse.documentMetadata();
        System.out.println("The number of pages in the document is " +
documentMetadata.pages());

    } catch (TextractException | FileNotFoundException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

Erkennen Sie Text aus einem Dokument, das sich in einem Amazon S3-Bucket befindet.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textract.model.Block;
```

```
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
import software.amazon.awssdk.services.textract.model.TextractException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectDocumentTextS3 {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <docName>\s

            Where:
                bucketName - The name of the Amazon S3 bucket that contains the
document.\s

                docName - The document name (must be an image, i.e., book.png).

\s

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String docName = args[1];
        Region region = Region.US_WEST_2;
        TextractClient textractClient = TextractClient.builder()
            .region(region)
            .build();

        detectDocTextS3(textractClient, bucketName, docName);
        textractClient.close();
    }
}
```

```
public static void detectDocTextS3(TextractClient textractClient, String
bucketName, String docName) {
    try {
        S3Object s3Object = S3Object.builder()
            .bucket(bucketName)
            .name(docName)
            .build();

        // Create a Document object and reference the s3Object instance.
        Document myDoc = Document.builder()
            .s3Object(s3Object)
            .build();

        DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
            .document(myDoc)
            .build();

        DetectDocumentTextResponse textResponse =
textractClient.detectDocumentText(detectDocumentTextRequest);
        for (Block block : textResponse.blocks()) {
            System.out.println("The block type is " +
block.blockType().toString());
        }

        DocumentMetadata documentMetadata = textResponse.documentMetadata();
        System.out.println("The number of pages in the document is " +
documentMetadata.pages());

    } catch (TextractException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Weitere API-Informationen finden Sie unter [DetectDocumentText](#) in der APIAWS SDK for Java 2.x -Referenz für .



## Starten der asynchronen Analyse eines Dokuments

Das folgende Codebeispiel zeigt, wie Sie die asynchrone Analyse eines Dokuments mit Amazon Textract starten.

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.StartDocumentAnalysisRequest;
import software.amazon.awssdk.services.textract.model.DocumentLocation;
import software.amazon.awssdk.services.textract.model.TextractException;
import software.amazon.awssdk.services.textract.model.StartDocumentAnalysisResponse;
import software.amazon.awssdk.services.textract.model.GetDocumentAnalysisRequest;
import software.amazon.awssdk.services.textract.model.GetDocumentAnalysisResponse;
import software.amazon.awssdk.services.textract.model.FeatureType;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StartDocumentAnalysis {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <docName>\s

                Where:
```

```
        bucketName - The name of the Amazon S3 bucket that contains the
document.\s
        docName - The document name (must be an image, for example,
book.png).\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String docName = args[1];
    Region region = Region.US_WEST_2;
    TextractClient textractClient = TextractClient.builder()
        .region(region)
        .build();

    String jobId = startDocAnalysisS3(textractClient, bucketName, docName);
    System.out.println("Getting results for job " + jobId);
    String status = getJobResults(textractClient, jobId);
    System.out.println("The job status is " + status);
    textractClient.close();
}

public static String startDocAnalysisS3(TextractClient textractClient, String
bucketName, String docName) {
    try {
        List<FeatureType> myList = new ArrayList<>();
        myList.add(FeatureType.TABLES);
        myList.add(FeatureType.FORMS);

        S3Object s3Object = S3Object.builder()
            .bucket(bucketName)
            .name(docName)
            .build();

        DocumentLocation location = DocumentLocation.builder()
            .s3Object(s3Object)
            .build();

        StartDocumentAnalysisRequest documentAnalysisRequest =
StartDocumentAnalysisRequest.builder()
            .documentLocation(location)
```

```
        .featureTypes(myList)
        .build();

        StartDocumentAnalysisResponse response =
textextractClient.startDocumentAnalysis(documentAnalysisRequest);

        // Get the job ID
        String jobId = response.jobId();
        return jobId;

    } catch (TextractException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

private static String getJobResults(TextractClient textractClient, String jobId)
{
    boolean finished = false;
    int index = 0;
    String status = "";

    try {
        while (!finished) {
            GetDocumentAnalysisRequest analysisRequest =
GetDocumentAnalysisRequest.builder()
                .jobId(jobId)
                .maxResults(1000)
                .build();

            GetDocumentAnalysisResponse response =
textextractClient.getDocumentAnalysis(analysisRequest);
            status = response.jobStatus().toString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(index + " status is: " + status);
                Thread.sleep(1000);
            }
            index++;
        }
    }
}
```

```
        return status;

    } catch (InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Weitere API-Informationen finden Sie unter [StartDocumentAnalysis](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Amazon TranscribeBeispiele mit SDK for Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen durchführen und gängige Szenarien implementieren, indem Sie die AWS SDK for Java 2.x mit Amazon Transcribe verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- [Aktionen](#)
- [Szenarien](#)

## Aktionen

### Auflisten von Transkriptionsaufträgen

Das folgende Codebeispiel zeigt, wie AmazonAmazon TranscribeTranskriptionsaufträge aufgelistet werden.

## SDK für Java 2.x

 Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public class ListTranscriptionJobs {
    public static void main(String[] args) {
        TranscribeClient transcribeClient = TranscribeClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listTranscriptionJobs(transcribeClient);
    }

    public static void listTranscriptionJobs(TranscribeClient transcribeClient)
    {
        ListTranscriptionJobsRequest listJobsRequest =
        ListTranscriptionJobsRequest.builder()
            .build();

        transcribeClient.listTranscriptionJobsPaginator(listJobsRequest).stream()
            .flatMap(response -> response.transcriptionJobSummaries().stream())
            .forEach(jobSummary -> {
                System.out.println("Job Name: " +
                jobSummary.transcriptionJobName());
                System.out.println("Job Status: " +
                jobSummary.transcriptionJobStatus());
                System.out.println("Output Location: " +
                jobSummary.outputLocationType());
                // Add more information as needed

                // Retrieve additional details for the job if necessary
                GetTranscriptionJobResponse jobDetails =
                transcribeClient.getTranscriptionJob(
                    GetTranscriptionJobRequest.builder()
                        .transcriptionJobName(jobSummary.transcriptionJobName())
                        .build());
            });
    }
}
```

```
        // Display additional details
        System.out.println("Language Code: " +
jobDetails.transcriptionJob().languageCode());
        System.out.println("Media Format: " +
jobDetails.transcriptionJob().mediaFormat());
        // Add more details as needed

        System.out.println("-----");
    });
}
}
```

- Weitere API-Informationen finden Sie unter [ListTranscriptionJobs](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Einen Transkriptionsauftrag starten

Das folgende Codebeispiel zeigt, wie Sie einen Amazon Transcribe-Transkriptionsauftrag starten.

SDK für Java 2.x

### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
        throws URISyntaxException, ExecutionException, InterruptedException,
LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
            .credentialsProvider(getCredentials())
            .region(REGION)
            .build();
    }
}
```

```
        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
    new AudioStreamPublisher(getStreamFromMic()),
    getResponseHandler());

    result.get();
    client.close();
}

private static InputStream getStreamFromMic() throws LineUnavailableException {

    // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
    int sampleRate = 16000;
    AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
    DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

    if (!AudioSystem.isLineSupported(info)) {
        System.out.println("Line not supported");
        System.exit(0);
    }

    TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
    line.open(format);
    line.start();

    InputStream audioStream = new AudioInputStream(line);
    return audioStream;
}

private static AwsCredentialsProvider getCredentials() {
    return DefaultCredentialsProvider.create();
}

private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
    return StartStreamTranscriptionRequest.builder()
        .languageCode(LanguageCode.EN_US.toString())
        .mediaEncoding(MediaEncoding.PCM)
        .mediaSampleRateHertz(mediaSampleRateHertz)
        .build();
}

private static StartStreamTranscriptionResponseHandler getResponseHandler() {
    return StartStreamTranscriptionResponseHandler.builder()

```

```

        .onResponse(r -> {
            System.out.println("Received Initial response");
        })
        .onError(e -> {
            System.out.println(e.getMessage());
            StringWriter sw = new StringWriter();
            e.printStackTrace(new PrintWriter(sw));
            System.out.println("Error Occurred: " + sw.toString());
        })
        .onComplete(() -> {
            System.out.println("=== All records stream successfully ===");
        })
        .subscriber(event -> {
            List<Result> results = ((TranscriptEvent)
event).transcript().results();
            if (results.size() > 0) {
                if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

System.out.println(results.get(0).alternatives().get(0).transcript());
                }
            }
        })
        .build();
    }

    private InputStream getStreamFromFile(String audioFileName) {
        try {
            File inputFile = new
File(getClass().getClassLoader().getResource(audioFileName).getFile());
            InputStream audioStream = new FileInputStream(inputFile);
            return audioStream;
        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }
    }

```



```

@Override
public void subscribe(Subscriber<? super AudioStream> s) {

    if (this.currentSubscription == null) {
        this.currentSubscription = new SubscriptionImpl(s, inputStream);
    } else {
        this.currentSubscription.cancel();
        this.currentSubscription = new SubscriptionImpl(s, inputStream);
    }
    s.onSubscribe(currentSubscription);
}

}

public static class SubscriptionImpl implements Subscription {
    private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
    private final Subscriber<? super AudioStream> subscriber;
    private final InputStream inputStream;
    private ExecutorService executor = Executors.newFixedThreadPool(1);
    private AtomicLong demand = new AtomicLong(0);

    SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
        this.subscriber = s;
        this.inputStream = inputStream;
    }

    @Override
    public void request(long n) {
        if (n <= 0) {
            subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
        }

        demand.getAndAdd(n);

        executor.submit(() -> {
            try {
                do {
                    ByteBuffer audioBuffer = getNextEvent();
                    if (audioBuffer.remaining() > 0) {
                        AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                        subscriber.onNext(audioEvent);
                    }
                } while (demand.get() > 0);
            } catch (Exception e) {
                subscriber.onError(e);
            }
        });
    }
}

```

```
        } else {
            subscriber.onComplete();
            break;
        }
    } while (demand.decrementAndGet() > 0);
} catch (Exception e) {
    subscriber.onError(e);
}
});
}

@Override
public void cancel() {
    executor.shutdown();
}

private ByteBuffer getNextEvent() {
    ByteBuffer audioBuffer = null;
    byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

    int len = 0;
    try {
        len = inputStream.read(audioBytes);

        if (len <= 0) {
            audioBuffer = ByteBuffer.allocate(0);
        } else {
            audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
        }
    } catch (IOException e) {
        throw new UncheckedIOException(e);
    }

    return audioBuffer;
}

private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
    return AudioEvent.builder()
        .audioChunk(SdkBytes.fromByteBuffer(bb))
        .build();
}
}
}
```

- Weitere API-Informationen finden Sie unter [StartTranscriptionJob](#) in der APIAWS SDK for Java 2.x -Referenz für .

## Szenarien

### Audio transkribieren und Auftragsdaten abrufen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Starten Sie einen Transkriptionsauftrag mit Amazon Transcribe.
- Warten Sie, bis der -Auftrag abgeschlossen wurde.
- Ermitteln Sie die URI, unter der das Transkript gespeichert ist.

Weitere Informationen finden Sie unter [Erste Schritte mit Amazon Transcribe](#).

### SDK für Java 2.x

#### Note

Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

### Transkribiert eine PCM-Datei.

```
/**
 * To run this AWS code example, ensure that you have set up your development
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class TranscribeStreamingDemoFile {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;
```

```
public static void main(String args[]) throws ExecutionException,
InterruptedException {

    final String USAGE = "\n" +
        "Usage:\n" +
        "    <file> \n\n" +
        "Where:\n" +
        "    file - the location of a PCM file to transcribe. In this
example, ensure the PCM file is 16 hertz (Hz). \n";

    if (args.length != 1) {
        System.out.println(USAGE);
        System.exit(1);
    }

    String file = args[0];
    client = TranscribeStreamingAsyncClient.builder()
        .region(REGION)
        .build();

    CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
    new AudioStreamPublisher(getStreamFromFile(file)),
    getResponseHandler());

    result.get();
    client.close();
}

private static InputStream getStreamFromFile(String file) {
    try {
        File inputFile = new File(file);
        InputStream audioStream = new FileInputStream(inputFile);
        return audioStream;

    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
}

private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
    return StartStreamTranscriptionRequest.builder()
        .languageCode(LanguageCode.EN_US)
```

```

        .mediaEncoding(MediaEncoding.PCM)
        .mediaSampleRateHertz(mediaSampleRateHertz)
        .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
            .onResponse(r -> {
                System.out.println("Received Initial response");
            })
            .onError(e -> {
                System.out.println(e.getMessage());
                StringWriter sw = new StringWriter();
                e.printStackTrace(new PrintWriter(sw));
                System.out.println("Error Occurred: " + sw.toString());
            })
            .onComplete(() -> {
                System.out.println("=== All records stream successfully ===");
            })
            .subscriber(event -> {
                List<Result> results = ((TranscriptEvent)
event).transcript().results();
                if (results.size() > 0) {
                    if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {
                        System.out.println(results.get(0).alternatives().get(0).transcript());
                    }
                }
            })
            .build();
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

```

```

        if (this.currentSubscription == null) {
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        } else {
            this.currentSubscription.cancel();
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        }
        s.onSubscribe(currentSubscription);
    }
}

public static class SubscriptionImpl implements Subscription {
    private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
    private final Subscriber<? super AudioStream> subscriber;
    private final InputStream inputStream;
    private ExecutorService executor = Executors.newFixedThreadPool(1);
    private AtomicLong demand = new AtomicLong(0);

    SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
    {
        this.subscriber = s;
        this.inputStream = inputStream;
    }

    @Override
    public void request(long n) {
        if (n <= 0) {
            subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
        }

        demand.getAndAdd(n);

        executor.submit(() -> {
            try {
                do {
                    ByteBuffer audioBuffer = getNextEvent();
                    if (audioBuffer.remaining() > 0) {
                        AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                        subscriber.onNext(audioEvent);
                    } else {
                        subscriber.onComplete();
                        break;
                    }
                }
            }
        });
    }
}

```

```
        } while (demand.decrementAndGet() > 0);
    } catch (Exception e) {
        subscriber.onError(e);
    }
});
}

@Override
public void cancel() {
    executor.shutdown();
}

private ByteBuffer getNextEvent() {
    ByteBuffer audioBuffer = null;
    byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

    int len = 0;
    try {
        len = inputStream.read(audioBytes);

        if (len <= 0) {
            audioBuffer = ByteBuffer.allocate(0);
        } else {
            audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
        }
    } catch (IOException e) {
        throw new UncheckedIOException(e);
    }

    return audioBuffer;
}

private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
    return AudioEvent.builder()
        .audioChunk(SdkBytes.fromByteBuffer(bb))
        .build();
}
}
}
```

Transkribiert Streaming-Audio vom Mikrofon Ihres Computers.

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
        throws URISyntaxException, ExecutionException, InterruptedException,
        LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
            .credentialsProvider(getCredentials())
            .region(REGION)
            .build();

        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
    new AudioStreamPublisher(getStreamFromMic()),
    getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
```



```
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
            .languageCode(LanguageCode.EN_US.toString())
            .mediaEncoding(MediaEncoding.PCM)
            .mediaSampleRateHertz(mediaSampleRateHertz)
            .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
            .onResponse(r -> {
                System.out.println("Received Initial response");
            })
            .onError(e -> {
                System.out.println(e.getMessage());
                StringWriter sw = new StringWriter();
                e.printStackTrace(new PrintWriter(sw));
                System.out.println("Error Occurred: " + sw.toString());
            })
            .onComplete(() -> {
                System.out.println("=== All records stream successfully ===");
            })
            .subscriber(event -> {
                List<Result> results = ((TranscriptEvent)
event).transcript().results();
                if (results.size() > 0) {
                    if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

System.out.println(results.get(0).alternatives().get(0).transcript());
                    }
                }
            })
            .build();
    }

    private InputStream getStreamFromFile(String audioFileName) {
        try {
            File inputFile = new
File(getClass().getClassLoader().getResource(audioFileName).getFile());
```

```
        InputStream audioStream = new FileInputStream(inputFile);
        return audioStream;
    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
}

private static class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;
    private static Subscription currentSubscription;

    private AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {

        if (this.currentSubscription == null) {
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        } else {
            this.currentSubscription.cancel();
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        }
        s.onSubscribe(currentSubscription);
    }
}

public static class SubscriptionImpl implements Subscription {
    private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
    private final Subscriber<? super AudioStream> subscriber;
    private final InputStream inputStream;
    private ExecutorService executor = Executors.newFixedThreadPool(1);
    private AtomicLong demand = new AtomicLong(0);

    SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
    {
        this.subscriber = s;
        this.inputStream = inputStream;
    }

    @Override
    public void request(long n) {
        if (n <= 0) {
```

```
        subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
    }

    demand.getAndAdd(n);

    executor.submit(() -> {
        try {
            do {
                ByteBuffer audioBuffer = getNextEvent();
                if (audioBuffer.remaining() > 0) {
                    AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                    subscriber.onNext(audioEvent);
                } else {
                    subscriber.onComplete();
                    break;
                }
            } while (demand.decrementAndGet() > 0);
        } catch (Exception e) {
            subscriber.onError(e);
        }
    });
}

@Override
public void cancel() {
    executor.shutdown();
}

private ByteBuffer getNextEvent() {
    ByteBuffer audioBuffer = null;
    byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

    int len = 0;
    try {
        len = inputStream.read(audioBytes);

        if (len <= 0) {
            audioBuffer = ByteBuffer.allocate(0);
        } else {
            audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
        }
    } catch (IOException e) {
```

```
        throw new UncheckedIOException(e);
    }

    return audioBuffer;
}

private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
    return AudioEvent.builder()
        .audioChunk(SdkBytes.fromByteBuffer(bb))
        .build();
}
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - [GetTranscriptionJob](#)
  - [StartTranscriptionJob](#)

## Serviceübergreifende Beispiele mit SDK for Java 2.x

Die folgenden Beispielanwendungen verwenden die AWS SDK for Java 2.x , um über mehrere hinweg zu arbeiten AWS-Services.

Serviceübergreifende Beispiele zielen auf eine fortgeschrittene Erfahrung ab, um Sie beim Erstellen von Anwendungen zu unterstützen.

### Beispiele

- [Erstellen Sie eine Anwendung zum Senden von Daten an eine DynamoDB-Tabelle](#)
- [Erstellen Sie einen Amazon Lex-Chatbot, um Ihre Website-Besucher einzubeziehen](#)
- [Erstellen einer Publish- und Abonnement-Anwendung, die Nachrichten übersetzt](#)
- [Erstellen einer Webanwendung, die Nachrichten mit Amazon SQS sendet und abrufft](#)
- [Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können](#)
- [Erstellen einer Webanwendung zur Verfolgung von DynamoDB-Daten](#)
- [Erstellen eines Amazon-Redshift-Element-Trackers](#)
- [Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben](#)

- [Erstellen einer Anwendung, die Kundenfeedback analysiert und Audio generiert](#)
- [Erkennen von PSA in Bildern mit Amazon Rekognition mithilfe eines - AWS SDK](#)
- [Erkennen von Objekten in Bildern mit Amazon Rekognition mithilfe eines AWS -SDK](#)
- [Erkennen von Personen und Objekten in einem Video mit Amazon Rekognition mithilfe eines AWS -SDK](#)
- [Veröffentlichen von Amazon SNS-Nachrichten in Amazon SQS-Warteschlangen mithilfe eines AWS -SDK](#)
- [Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion](#)
- [Verwenden von Step Functions, um Lambda-Funktionen aufzurufen](#)
- [Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion](#)

## Erstellen Sie eine Anwendung zum Senden von Daten an eine DynamoDB-Tabelle

### SDK für Java 2.x

Zeigt, wie man eine dynamische Webanwendung erstellt, die Daten über die Amazon-DynamoDB-Java-API übermittelt und eine Textnachricht über die Amazon Simple Notification Service Java API sendet.

Vollständiger Quellcode und Anweisungen zum Einrichten und Ausführen finden Sie im vollständigen Beispiel auf [GitHub](#).

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SNS

## Erstellen Sie einen Amazon Lex-Chatbot, um Ihre Website-Besucher einzubeziehen

### SDK für Java 2.x

Zeigt, wie Sie die Amazon Lex-API verwenden, um einen Chatbot innerhalb einer Webanwendung zu erstellen, um Ihre Website-Besucher einzubeziehen.

Vollständiger Quellcode und Anweisungen zum Einrichten und Ausführen finden Sie im vollständigen Beispiel auf [GitHub](#).

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

## Erstellen einer Publish- und Abonnement-Anwendung, die Nachrichten übersetzt

SDK für Java 2.x

Zeigt, wie man die Java-API für Amazon Simple Notification Service verwendet, um eine Webanwendung zu erstellen, die über Abonnement- und Veröffentlichungsfunktionen verfügt. Darüber hinaus übersetzt diese Beispielanwendung auch Nachrichten.

Vollständiger Quellcode und Anweisungen zum Einrichten und Ausführen finden Sie im vollständigen Beispiel auf [GitHub](#).

Vollständiger Quellcode und Anweisungen zum Einrichten und Ausführen des Beispiels, das die Java Async API verwendet, finden Sie im vollständigen Beispiel auf [GitHub](#).

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon Translate

## Erstellen einer Webanwendung, die Nachrichten mit Amazon SQS sendet und abrufen

SDK für Java 2.x

Zeigt, wie Sie die Amazon SQS-API verwenden, um eine Spring-REST-API zu entwickeln, die Nachrichten sendet und abrufen.

Vollständiger Quellcode und Anweisungen zum Einrichten und Ausführen finden Sie im vollständigen Beispiel auf [GitHub](#).

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon SQS

## Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können

SDK für Java 2.x

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.

Vollständiger Quellcode und Anweisungen zum Einrichten und Ausführen finden Sie im vollständigen Beispiel auf [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

SDK für Java 2.x

Zeigt, wie man die Amazon-DynamoDB-API verwendet, um eine dynamische Webanwendung zu erstellen, die DynamoDB-Arbeitsdaten verfolgt.

Vollständiger Quellcode und Anweisungen zum Einrichten und Ausführen finden Sie im vollständigen Beispiel auf [GitHub](#).

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SES

## Erstellen eines Amazon-Redshift-Element-Trackers

SDK für Java 2.x

Zeigt, wie eine Webanwendung erstellt wird, die in einer Amazon-Redshift-Datenbank gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Vollständiger Quellcode und Anweisungen zum Einrichten einer Spring-REST-API, die Amazon-Redshift-Daten abfragt, und zur Verwendung durch eine React-Anwendung finden Sie im vollständigen Beispiel auf [GitHub](#).

In diesem Beispiel verwendete Dienste

- Amazon Redshift
- Amazon SES

## Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

SDK für Java 2.x

Zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente, die in einer Amazon RDS-Datenbank gespeichert sind, verfolgt und darüber berichtet.

Vollständiger Quellcode und Anweisungen zum Einrichten einer Spring-REST-API, die Amazon-Aurora-Serverless-Daten abfragt, und zur Verwendung durch eine React-Anwendung finden Sie im vollständigen Beispiel auf [GitHub](#).

Vollständiger Quellcode und Anweisungen zum Einrichten und Ausführen eines Beispiels, das die JDBC-API verwendet, finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS



- Amazon RDS Data Service
- Amazon SES

## Erstellen einer Anwendung, die Kundenfeedback analysiert und Audio generiert

### SDK für Java 2.x

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Quellcode und Bereitstellungsanweisungen finden Sie im Projekt unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

## Erkennen von PSA in Bildern mit Amazon Rekognition mithilfe eines - AWS SDK

### SDK für Java 2.x

Zeigt, wie Sie eine - AWS Lambda Funktion erstellen, die Bilder mit persönlicher Schutzausrüstung erkennt.

Vollständiger Quellcode und Anweisungen zum Einrichten und Ausführen finden Sie im vollständigen Beispiel auf [GitHub](#).

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

## Erkennen von Objekten in Bildern mit Amazon Rekognition mithilfe eines AWS -SDK

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-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).

Vollständiger Quellcode und Anweisungen zum Einrichten und Ausführen finden Sie im vollständigen Beispiel auf [GitHub](#).

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Erkennen von Personen und Objekten in einem Video mit Amazon Rekognition mithilfe eines AWS -SDK

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Gesichter und Objekte in Videos erkennt, 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).

Vollständiger Quellcode und Anweisungen zum Einrichten und Ausführen finden Sie im vollständigen Beispiel auf [GitHub](#).

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Veröffentlichen von Amazon SNS-Nachrichten in Amazon SQS-Warteschlangen mithilfe eines AWS -SDK

SDK für Java 2.x

Zeigt das Messaging mit Themen und Warteschlangen mithilfe von Amazon Simple Notification Service (Amazon SNS) und Amazon Simple Queue Service (Amazon SQS).

Vollständiger Quellcode und Anweisungen, die das Messaging mit Themen und Warteschlangen in Amazon SNS und Amazon SQS demonstrieren, finden Sie im vollständigen Beispiel auf [GitHub](#).

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon SQS

## Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion

SDK für Java 2.x

Zeigt, wie eine - AWS Lambda Funktion mithilfe der Lambda-Java-Laufzeit-API erstellt wird. In diesem Beispiel werden verschiedene - AWS Services aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine Lambda-Funktion erstellt, die von Amazon API Gateway aufgerufen wird und eine Amazon-DynamoDB-Tabelle nach Arbeitsjubiläen durchsucht und Amazon Simple Notification Service (Amazon SNS) verwendet,

um eine Textnachricht an Ihre Mitarbeiter zu senden, die ihnen zu ihrem einjährigen Jubiläum gratuliert.

Vollständiger Quellcode und Anweisungen zum Einrichten und Ausführen finden Sie im vollständigen Beispiel auf [GitHub](#).

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

## Verwenden von Step Functions, um Lambda-Funktionen aufzurufen

SDK für Java 2.x

Zeigt, wie Sie einen AWS -Serverless-Workflow mithilfe von AWS Step Functions und erstellen AWS SDK for Java 2.x. Jeder Workflow-Schritt wird mithilfe einer - AWS Lambda Funktion implementiert.

Vollständiger Quellcode und Anweisungen zum Einrichten und Ausführen finden Sie im vollständigen Beispiel auf [GitHub](#).

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

## Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

SDK für Java 2.x

Zeigt, wie Sie ein EventBridge geplantes Amazon-Ereignis erstellen, das eine - AWS Lambda Funktion aufruft. Konfigurieren Sie so EventBridge , dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine

Lambda-Funktion mithilfe der Lambda-Java-Laufzeit-API. In diesem Beispiel werden verschiedene - AWS Services aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Vollständiger Quellcode und Anweisungen zum Einrichten und Ausführen finden Sie im vollständigen Beispiel auf [GitHub](#).

In diesem Beispiel verwendete Dienste

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

# Sicherheit für die AWS SDK for Java

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 geteilte Verantwortung zwischen 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, die alle in der AWS Cloud angebotenen Services ausführt, und für die Bereitstellung von Services, die Sie sicher nutzen können. Unsere Sicherheitsverantwortung hat bei höchster Priorität AWS, und die Effektivität unserer Sicherheit wird regelmäßig von externen Prüfern im Rahmen der [AWS -Compliance-Programme](#) getestet und überprüft.

**Sicherheit in der Cloud** – Ihre Verantwortung wird durch den AWS Service bestimmt, den Sie verwenden, sowie durch andere Faktoren wie die Vertraulichkeit Ihrer Daten, die Anforderungen Ihrer Organisation und die geltenden Gesetze und Vorschriften.

Dieses AWS Produkt oder dieser Service folgt dem [Modell der geteilten Verantwortung](#) durch die spezifischen Amazon Web Services (AWS)-Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der [AWS Seite zur Servicesicherheit](#) und [AWS in den Services, die im Rahmen von AWS Compliance-Bemühungen nach Compliance-Programmen enthalten sind](#).

## Themen

- [Datenschutz in AWS SDK for Java 2.x](#)
- [Arbeiten mit TLS im SDK for Java](#)
- [Identitäts- und Zugriffsverwaltung](#)
- [Compliance-Validierung für dieses AWS Produkt oder diesen Service](#)
- [Ausfallsicherheit für dieses AWS Produkt oder diesen Service](#)
- [Infrastruktursicherheit für dieses AWS Produkt oder diesen Service](#)

## Datenschutz in AWS SDK for Java 2.x

Das AWS [Modell der geteilten Verantwortung](#)Modell gilt für den Datenschutz in AWS SDK for Java. Wie in diesem Modell beschrieben, AWS ist für den Schutz der globalen Infrastruktur

verantwortlich, die alle ausführt 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 Ihnen, -Anmeldeinformationen zu schützen AWS-Konto und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einzurichten. 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).
- Verwenden Sie SSL/TLS für die Kommunikation mit - AWS Ressourcen. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API- und Benutzeraktivitätsprotokollierung mit ein AWS CloudTrail.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen Standardsicherheitskontrollen in 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 auf AWS über eine Befehlszeilenschnittstelle oder eine API FIPS-140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter [Federal Information Processing Standard \(FIPS\) 140-2](#).

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 SDK for Java oder anderen AWS-Services unter Verwendung der Konsole, API AWS CLI oder AWS SDKs arbeiten. 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.

## Arbeiten mit TLS im SDK for Java

Die AWS SDK for Java nutzt die TLS-Funktionen der zugrunde liegenden Java-Plattform. In diesem Thema zeigen wir Beispiele für die OpenJDK-Implementierung, die von [Amazon Corretto 17](#) verwendet wird.

Um mit arbeiten zu können AWS-Services, muss das zugrunde liegende JDK eine Mindestversion von TLS 1.2 unterstützen, aber TLS 1.3 wird empfohlen.

Benutzer sollten in der Dokumentation der Java-Plattform, die sie mit dem SDK verwenden, nachsehen, welche TLS-Versionen standardmäßig aktiviert sind und wie bestimmte TLS-Versionen aktiviert und deaktiviert werden.

### So überprüfen Sie die Informationen zur TLS-Version

Mit OpenJDK zeigt der folgende Code die Verwendung von [SSLContext](#), um zu drucken, welche TLS/SSL-Versionen unterstützt werden.

```
System.out.println(Arrays.toString(SSLContext.getDefault().getSupportedSSLParameters().getProtocols()));
```

Beispielsweise erzeugt Amazon Corretto 17 (OpenJDK ) die folgende Ausgabe.

```
[TLSv1.3, TLSv1.2, TLSv1.1, TLSv1, SSLv3, SSLv2Hello]
```

Um den SSL-Handshake in Aktion zu sehen und welche Version von TLS verwendet wird, können Sie die Systemeigenschaft `javax.net.debug` verwenden.

Führen Sie beispielsweise Java-Anwendungen aus, die TLS verwenden.

```
java app.jar -Djavax.net.debug=ssl:handshake
```

Die Anwendung protokolliert den SSL-Handshake ähnlich dem folgenden.

```
...
javax.net.ssl|DEBUG|10|main|2022-12-23 13:53:12.221 EST|ClientHello.java:641|Produced
ClientHello handshake message (
"ClientHello": {
  "client version"      : "TLSv1.2",
```



```
...
javax.net.ssl|DEBUG|10|main|2022-12-23 13:53:12.295 EST|ServerHello.java:888|Consuming
  ServerHello handshake message (
    "ServerHello": {
      "server version"      : "TLSv1.2",
    }
  )
...
```

## Erzwingen einer TLS-Mindestversion

Das SDK for Java bevorzugt immer die neueste TLS-Version, die von der Plattform und dem Service unterstützt wird. Wenn Sie eine bestimmte TLS-Mindestversion erzwingen möchten, lesen Sie die Dokumentation Ihrer Java-Plattform.

Für OpenJDK-basierte JVMs können Sie die Systemeigenschaft `jdk.tls.client.protocols` verwenden.

Wenn Sie beispielsweise möchten, dass SDK-Service-Clients in Ihrer Anwendung TLS 1.2 verwenden, obwohl TLS 1.3 verfügbar ist, geben Sie die folgende Systemeigenschaft an.

```
java app.jar -Djdk.tls.client.protocols=TLSv1.2
```

## AWS Upgrade von API-Endpunkten auf TLS 1.2

In diesem [Blogbeitrag](#) finden Sie Informationen über API AWS -Endpunkte, die für die Mindestversion zu TLS 1.2 verschoben werden.

## Identitäts- und Zugriffsverwaltung

AWS Identity and Access Management (IAM) ist ein AWS-Service , mit dem ein Administrator den Zugriff auf - AWS Ressourcen sicher steuern kann. IAM-Administratoren steuern, wer für die Nutzung von - AWS Ressourcen authentifiziert (angemeldet) und autorisiert (im Besitz von Berechtigungen) werden kann. IAM ist ein AWS-Service , den Sie ohne zusätzliche Kosten verwenden können.

Themen

- [Zielgruppe](#)
- [Authentifizierung mit Identitäten](#)
- [Verwalten des Zugriffs mit Richtlinien](#)
- [Funktionsweise AWS-Services von mit IAM](#)

- [Fehlerbehebung für AWS Identität und Zugriff](#)

## Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, unterscheidet sich je nach Ihrer Arbeit in AWS.

**Service-Benutzer** – Wenn Sie AWS-Services zur Ausführung von Aufgaben verwenden, stellt Ihnen Ihr Administrator die Anmeldeinformationen und Berechtigungen bereit, die Sie benötigen. Wenn Sie für Ihre Arbeit weitere AWS Funktionen ausführen, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Funktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anfordern müssen. Wenn Sie nicht auf ein Feature zugreifen können, finden Sie weitere Informationen unter [Fehlerbehebung für AWS Identität und Zugriff](#) oder im Benutzerhandbuch des von AWS-Service Ihnen verwendeten .

**Service-Administrator** – Wenn Sie in Ihrem Unternehmen für AWS Ressourcen verantwortlich sind, haben Sie wahrscheinlich vollständigen Zugriff auf AWS. Es ist Ihre Aufgabe, zu bestimmen, auf welche AWS Funktionen und Ressourcen Ihre Service-Benutzer zugreifen sollen. Sie müssen dann Anträge an Ihren IAM-Administrator stellen, um die Berechtigungen Ihrer Servicenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen darüber, wie Ihr Unternehmen IAM mit verwenden kann AWS, finden Sie im Benutzerhandbuch der , die AWS-Service Sie verwenden.

**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 , die AWS-Service Sie verwenden.

## Authentifizierung mit Identitäten

Die Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten bei anmelden. Sie müssen als Root-Benutzer des AWS-Kontos, als IAM-Benutzer oder durch Übernahme einer IAM-Rolle authentifiziert (bei angemeldet AWS) sein.

Sie können sich bei AWS als Verbundidentität anmelden, indem Sie Anmeldeinformationen verwenden, die über eine Identitätsquelle bereitgestellt werden. AWS IAM Identity Center (IAM Identity Center)-Benutzer, die Single-Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für Verbundidentitäten. Wenn Sie

sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen Identitätsverbund eingerichtet. Wenn Sie AWS über einen Verbund auf zugreifen, übernehmen Sie indirekt eine Rolle.

Je nachdem, um welchen Benutzertyp es sich handelt, können Sie sich bei der AWS Management Console oder im - AWS Zugriffsportal anmelden. Weitere Informationen zur Anmeldung bei AWS finden Sie unter [So melden Sie sich bei Ihrem an AWS-Konto](#) im AWS-Anmeldung - Benutzerhandbuch.

Wenn Sie AWS programmgesteuert auf zugreifen, AWS stellt ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (Command Line Interface, CLI) bereit, um Ihre Anforderungen mithilfe Ihrer Anmeldeinformationen kryptografisch zu signieren. Wenn Sie keine AWS Tools verwenden, müssen Sie Anforderungen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode zum eigenständigen Signieren von Anforderungen finden Sie unter [Signieren von AWS API-Anforderungen](#) im IAM-Benutzerhandbuch.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen angeben. empfiehlt beispielsweise, AWS Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere Informationen finden Sie unter [Multi-Faktor-Authentifizierung](#) im AWS IAM Identity Center - Benutzerhandbuch und [Verwenden der Multi-Faktor-Authentifizierung \(MFA\) in AWS](#) im IAM-Benutzerhandbuch.

## AWS-Konto Root-Benutzer

Wenn Sie ein erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services und Ressourcen im Konto hat. Diese Identität wird als AWS-Konto Root-Benutzer bezeichnet und Sie melden sich mit der E-Mail-Adresse und dem Passwort an, mit denen Sie das Konto erstellt haben. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen und verwenden Sie diese, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter [Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern](#) im IAM-Benutzerhandbuch.

## Verbundidentität

Fordern Sie als bewährte Methode menschliche Benutzer, einschließlich Benutzer, die Administratorzugriff benötigen, auf, den Verbund mit einem Identitätsanbieter zu verwenden, um AWS-Services mithilfe temporärer Anmeldeinformationen auf zuzugreifen.

Eine Verbundidentität ist ein Benutzer aus Ihrem Unternehmensbenutzerverzeichnis, ein Web-Identitätsanbieter, die AWS Directory Service, das Identity-Center-Verzeichnis oder jeder Benutzer, der mit AWS-Services Anmeldeinformationen auf zugreift, die über eine Identitätsquelle bereitgestellt werden. Wenn Verbundidentitäten auf zugreifen AWS-Konten, übernehmen sie Rollen und die Rollen stellen temporäre Anmeldeinformationen bereit.

Für die zentrale Zugriffsverwaltung empfehlen wir Ihnen, AWS IAM Identity Center zu verwenden. Sie können Benutzer und Gruppen in IAM Identity Center erstellen oder eine Verbindung zu einer Gruppe von Benutzern und Gruppen in Ihrer eigenen Identitätsquelle herstellen und synchronisieren, um sie für alle Ihre AWS-Konten und Anwendungen zu verwenden. Informationen zu IAM Identity Center finden Sie unter [Was ist IAM Identity Center?](#) im AWS IAM Identity Center -Benutzerhandbuch.

## IAM-Benutzer und -Gruppen

Ein [IAM-Benutzer](#) ist eine Identität in Ihrem AWS-Konto mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter [Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern](#) im IAM-Benutzerhandbuch.

Eine [IAM-Gruppe](#) ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise einer Gruppe mit dem Namen IAMAdmins Berechtigungen zum Verwalten von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter [Erstellen eines IAM-Benutzers \(anstatt einer Rolle\)](#) im IAM-Benutzerhandbuch.

## IAM-Rollen

Eine [IAM-Rolle](#) ist eine Identität in Ihrem AWS-Konto mit bestimmten Berechtigungen. Sie ist einem IAM-Benutzer vergleichbar, ist aber nicht mit einer bestimmten Person verknüpft. Sie können vorübergehend eine IAM-Rolle in der übernehmen, AWS Management Console indem Sie die [Rollen](#)

[wechseln](#). Sie können eine Rolle übernehmen, indem Sie eine AWS CLI - oder AWS -API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter [Verwenden von IAM-Rollen](#) im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- **Verbundbenutzerzugriff** – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter [Erstellen von Rollen für externe Identitätsanbieter](#) im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter [Berechtigungssätze](#) im AWS IAM Identity Center -Benutzerhandbuch.
- **Temporäre IAM-Benutzerberechtigungen** – Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- **Kontoübergreifender Zugriff** – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Bei einigen können AWS-Services Sie jedoch eine Richtlinie direkt an eine Ressource anfügen (anstatt eine Rolle als Proxy zu verwenden). Informationen zu den Unterschieden zwischen Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter [So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien](#) im IAM-Benutzerhandbuch.
- **Serviceübergreifender Zugriff** – Einige AWS-Services verwenden Funktionen in anderen AWS-Services. Wenn Sie beispielsweise einen Aufruf in einem Service tätigen, führt dieser Service häufig Anwendungen in Amazon EC2 aus oder speichert Objekte in Amazon S3. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servic Rolle oder mit einer serviceverknüpften Rolle tun.
- **Forward Access Sessions (FAS)** – Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen in auszuführen AWS, gelten Sie als Prinzipal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service auslösen. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anforderung AWS-Service , Anfragen an nachgelagerte Services zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Service eine Anfrage erhält, für deren Abschluss Interaktionen

mit anderen AWS-Services oder -Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter [Zugriffssitzungen weiterleiten](#).

- **Servicerolle:** Eine Servicerolle ist eine [IAM-Rolle](#), die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter [Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service](#) im IAM-Benutzerhandbuch.
- **Serviceverknüpfte Rolle** – Eine serviceverknüpfte Rolle ist eine Art von Servicerolle, die mit einem verknüpft ist AWS-Service. Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Serviceverknüpfte Rollen werden in Ihrem angezeigt AWS-Konto und gehören dem Service. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.
- **Anwendungen, die auf Amazon EC2 ausgeführt werden** – Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2-Instance ausgeführt werden und - AWS CLI oder AWS -API-Anforderungen stellen. Das ist eher zu empfehlen, als Zugriffsschlüssel innerhalb der EC2-Instance zu speichern. Um einer EC2-Instance eine - AWS Rolle zuzuweisen und sie für alle ihre Anwendungen verfügbar zu machen, erstellen Sie ein Instance-Profil, das an die Instance angehängt ist. Ein Instance-Profil enthält die Rolle und ermöglicht, dass Programme, die in der EC2-Instance ausgeführt werden, temporäre Anmeldeinformationen erhalten. Weitere Informationen finden Sie unter [Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden](#) im IAM-Benutzerhandbuch.

Informationen dazu, wann Sie IAM-Rollen oder IAM-Benutzer verwenden sollten, finden Sie unter [Erstellen einer IAM-Rolle \(anstatt eines Benutzers\)](#) im IAM-Benutzerhandbuch.

## Verwalten des Zugriffs mit Richtlinien

Sie steuern den Zugriff in , AWS indem Sie Richtlinien erstellen und sie an AWS Identitäten oder Ressourcen anfügen. Eine Richtlinie ist ein Objekt in , AWS das, wenn es einer Identität oder Ressource zugeordnet wird, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anforderung stellt. Berechtigungen in den Richtlinien bestimmen, ob die Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden in AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und Inhalten von JSON-Richtliniendokumenten finden Sie unter [Übersicht über JSON-Richtlinien](#) im IAM-Benutzerhandbuch.

Administratoren können AWS JSON-Richtlinien verwenden, um festzulegen, wer Zugriff auf was hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen aus der AWS Management Console, der AWS CLI oder der AWS -API abrufen.

## Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter [Erstellen von IAM-Richtlinien](#) im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem anfügen können AWS-Konto. Verwaltete Richtlinien umfassen - AWS verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer eingebundenen Richtlinie wählen, finden Sie unter [Auswahl zwischen verwalteten und eingebundenen Richtlinien](#) im IAM-Benutzerhandbuch.

## Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien 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. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen



in einer ressourcenbasierten Richtlinie [einen Prinzipal angeben](#). Prinzipale können Konten, Benutzer, Rollen, Verbundbenutzer oder umfassen AWS-Services.

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

## Zugriffssteuerungslisten (ACLs)

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Services AWS WAF, die ACLs unterstützen. Weitere Informationen zu ACLs finden Sie unter [Zugriffssteuerungsliste \(ACL\) – Übersicht](#) (Access Control List) im Amazon-Simple-Storage-Service-Entwicklerhandbuch.

## Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger häufig verwendete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- **Berechtigungsgrenzen** – Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter [Berechtigungsgrenzen für IAM-Entitäten](#) im IAM-Benutzerhandbuch.
- **Service-Kontrollrichtlinien (SCPs)** – SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in angeben AWS Organizations. AWS Organizations ist ein Service zum Gruppieren und zentralen Verwalten mehrerer AWS-Konten, die Ihrem Unternehmen gehören. Wenn Sie innerhalb einer Organisation alle Features aktivieren, können Sie Service-Kontrollrichtlinien (SCPs) auf alle oder einzelne Ihrer Konten anwenden. Die SCP beschränkt Berechtigungen für Entitäten in Mitgliedskonten, einschließlich jeder Root-Benutzer des AWS-Kontos. Weitere Informationen zu Organizations und SCPs finden Sie unter [Funktionsweise von SCPs](#) im AWS Organizations -Benutzerhandbuch.



- **Sitzungsrichtlinien** – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter [Sitzungsrichtlinien](#) im IAM-Benutzerhandbuch.

## Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen dazu, wie AWS bestimmt, ob eine Anforderung zugelassen werden soll, wenn mehrere Richtlinientypen beteiligt sind, finden Sie unter [Logik zur Richtlinienbewertung](#) im IAM-Benutzerhandbuch.

## Funktionsweise AWS-Services von mit IAM

Einen Überblick über die AWS-Services Funktionsweise der meisten IAM-Funktionen finden Sie unter [-AWS Services, die mit IAM funktionieren](#) im IAM-Benutzerhandbuch.

Informationen zur Verwendung eines bestimmten AWS-Service mit IAM finden Sie im Abschnitt Sicherheit im Benutzerhandbuch des entsprechenden Services.

## Fehlerbehebung für AWS Identität und Zugriff

Verwenden Sie die folgenden Informationen, um häufige Probleme zu diagnostizieren und zu beheben, die beim Arbeiten mit AWS und IAM auftreten können.

### Themen

- [Ich bin nicht autorisiert, eine Aktion in auszuführen AWS](#)
- [Ich bin nicht autorisiert, iam durchzuführen:PassRole](#)
- [Ich möchte Personen außerhalb meines AWS-Konto Zugriff auf meine - AWS Ressourcen gewähren](#)

## Ich bin nicht autorisiert, eine Aktion in auszuführen 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 `aws:GetWidget`-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
aws:GetWidget on resource: my-example-widget
```

In diesem Fall muss die Richtlinie für den Benutzer `mateojackson` aktualisiert werden, damit er mit der `aws: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 autorisiert, iam durchzuführen: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 AWS übergeben zu können.

Einige AWS-Services ermöglichen es Ihnen, eine vorhandene Rolle an diesen Service zu übergeben, anstatt eine neue Servicerolle oder serviceverknüpfte 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 meines AWS-Konto Zugriff auf meine - AWS Ressourcen gewähren

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Im Fall von Services, die ressourcenbasierte Richtlinien oder Zugriffssteuerungslisten (Access Control Lists, ACLs) verwenden, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen dazu, ob diese Funktionen AWS unterstützt, finden Sie unter [Funktionsweise AWS-Services von mit IAM](#).
- Informationen zum Gewähren des Zugriffs auf Ihre AWS-Konten -Ressourcen in Ihrem Besitz finden Sie unter [Gewähren des Zugriffs für einen IAM-Benutzer in einem anderen AWS-Konto , das Sie besitzen](#) im IAM-Benutzerhandbuch.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre -Ressourcen gewähren AWS-Konten, finden Sie unter [Gewähren von Zugriff auf im AWS-Konten Besitz von Dritten](#) im IAM-Benutzerhandbuch.
- 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 [So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien](#) im IAM-Benutzerhandbuch.


## Compliance-Validierung für dieses AWS Produkt oder diesen Service

Informationen darüber, ob ein in den Geltungsbereich bestimmter Compliance-Programme AWS-Service fällt, finden Sie [AWS-Services unter im Geltungsbereich nach Compliance-Programm](#) und wählen Sie das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter [AWS Compliance-Programme](#)

Sie können Auditberichte von Drittanbietern mit herunterladen AWS Artifact. Weitere Informationen finden Sie unter [Herunterladen von Berichten unter AWS Artifact](#) .

Ihre Compliance-Verantwortung bei der Verwendung von AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Unterstützung der Compliance bereit:

- [Schnellstartanleitungen für Sicherheit und Compliance](#) – In diesen Bereitstellungsleitfäden werden Überlegungen zur Architektur erörtert und Schritte für die Bereitstellung von Basisumgebungen in bereitgestellt AWS , die sich auf Sicherheit und Compliance konzentrieren.
- [Architekturerstellung für HIPAA-Sicherheit und -Compliance in Amazon Web Services](#) – In diesem Whitepaper wird beschrieben, wie Unternehmen mithilfe AWS von HIPAA-berechtigte Anwendungen erstellen können.

 Note

Nicht alle AWS-Services sind HIPAA-berechtigt. Weitere Informationen finden Sie in der [Referenz für HIPAA-berechtigte Services](#).

- [AWS Compliance-Ressourcen](#) – Diese Sammlung von Arbeitsmappen und Leitfäden könnte für Ihre Branche und Ihren Standort gelten.
- [AWS Kunden-Compliance-Leitfäden](#) – Verstehen Sie das Modell der geteilten Verantwortung anhand der Compliance. Die Leitfäden fassen die bewährten Methoden zur Sicherung zusammen AWS-Services und ordnen die Leitlinien den Sicherheitskontrollen in mehreren Frameworks zu (einschließlich National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Officer (PCI) und International Organization for Standardization (ISO)).
- [Bewertung von Ressourcen mit Regeln](#) im -AWS Config Entwicklerhandbuch – Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- [AWS Security Hub](#) – Dies AWS-Service bietet einen umfassenden Überblick über Ihren Sicherheitsstatus innerhalb von AWS. Security Hub verwendet Sicherheitskontrollen, um Ihre AWS -Ressourcen zu bewerten und Ihre Einhaltung von Sicherheitsstandards und bewährten Methoden zu überprüfen. Eine Liste der unterstützten Services und Kontrollen finden Sie in der [Security-Hub-Steuerungsreferenz](#).
- [AWS Audit Manager](#) – Auf diese AWS-Service Weise können Sie Ihre AWS Nutzung kontinuierlich überprüfen, um den Umgang mit Risiken und die Einhaltung von Branchenstandards zu vereinfachen.

Dieses AWS Produkt oder dieser Service folgt dem [Modell der geteilten Verantwortung](#) durch die spezifischen Amazon Web Services (AWS)-Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der [AWS Seite zur Servicesicherheit](#) und [AWS in den Services, die im Rahmen von AWS Compliance-Bemühungen nach Compliance-Programmen enthalten sind](#).

## Ausfallsicherheit für dieses AWS Produkt oder diesen Service

Die AWS globale -Infrastruktur ist um AWS-Regionen und Availability Zones herum aufgebaut.

AWS-Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die mit einem Netzwerk mit niedriger Latenz, hohem Durchsatz und hoher Redundanz 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 geteilten Verantwortung](#) durch die spezifischen Amazon Web Services (AWS)-Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der [AWS Seite mit der Service-Sicherheitsdokumentation](#) und [AWS den Services, die im Rahmen der AWS Compliance-Bemühungen nach Compliance-Programmen enthalten sind](#).

## Infrastruktursicherheit für dieses AWS Produkt oder diesen Service

Dieses AWS Produkt oder dieser Service verwendet verwaltete Services und ist daher durch die AWS globale Netzwerksicherheit von geschützt. Informationen zu AWS Sicherheitservices und wie die Infrastruktur AWS schützt, finden Sie unter [AWS Cloud-Sicherheit](#). Informationen zum Entwerfen Ihrer AWS Umgebung mit den bewährten Methoden für die Infrastruktursicherheit finden Sie unter [Infrastrukturschutz](#) in Security Pillar AWS Well-Architected Framework.

Sie verwenden durch 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 geteilten Verantwortung](#) durch die spezifischen Amazon Web Services (AWS)-Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der [AWS Seite zur Servicesicherheit](#) und [AWS in den Services, die im Rahmen der AWS Compliance-Bemühungen nach Compliance-Programmen enthalten sind](#).

# Migrieren von Version 1.x zu 2.x des AWS SDK for Java

Die AWS SDK for Java 2.x ist eine wichtige Umschreibung der 1.x-Codebasis, die auf Java 8+ basiert. Es enthält viele Updates wie z. B. verbesserte Konsistenz, Benutzerfreundlichkeit und nachdrücklich umgesetzte Unveränderlichkeit. In diesem Abschnitt werden die wichtigsten Features beschrieben, die in Version 2.x neu sind, und es finden Sie Anleitungen zur Migration Ihres Codes zu Version 2.x von 1.x.

## Themen

- [Was ist neu in Version 2?](#)
- [step-by-step Migrationsanweisungen mit Beispiel](#)
- [Was unterscheidet sich zwischen AWS SDK for Java 1.x und 2.x?](#)
- [Verwenden Sie das SDK for Java 1.x und 2.x side-by-side](#)

## Was ist neu in Version 2?

- Sie können eigene HTTP-Clients konfigurieren. Siehe [HTTP-Transportkonfiguration](#).
- Asynchrone Clients verfügen über nicht blockierende E/A-Unterstützung und geben `CompletableFuture` Objekte zurück. Siehe [Asynchrone Programmierung](#).
- Operationen, die mehrere Seiten zurückgeben, verfügen über automatische paginierte Antworten. Auf diese Weise können Sie Ihren Code darauf konzentrieren, was mit der Antwort geschehen soll, ohne nach nachfolgenden Seiten suchen und diese abrufen zu müssen. Siehe [Paginierung](#).
- Die Leistung der Startzeit des SDK für - AWS Lambda Funktionen wurde verbessert. Weitere Informationen finden Sie unter Verbesserungen [der Leistung der Startzeit des SDK](#).
- Version 2.x unterstützt eine neue Kurzmethode zum Erstellen von Anfragen.

## Example

```
dynamoDbClient.putItem(request -> request.tableName(TABLE))
```

Weitere Informationen zu den neuen Funktionen und spezifische Codebeispiele finden Sie in den anderen Abschnitten dieses Handbuchs.

- [Schnellstart](#)

- [Einrichtung](#)
- [Codebeispiele für AWS SDK for Java 2.x](#)
- [Verwenden des SDK](#)
- [Sicherheit für die AWS SDK for Java](#)

## step-by-step Migrationsanweisungen mit Beispiel

Dieser Abschnitt enthält eine step-by-step Anleitung für die Migration Ihrer Anwendung, die derzeit das SDK für Java v1.x verwendet, zum SDK für Java 2.x. Der erste Teil bietet einen Überblick über die Schritte, gefolgt von einem detaillierten Beispiel für eine Migration.

Die hier beschriebenen Schritte beschreiben eine Migration eines normalen Anwendungsfalls, bei dem die Anwendung AWS-Services mithilfe von modellgesteuerten Service-Clients aufruft. Wenn Sie Code migrieren müssen, der übergeordnete APIs wie [S3 Transfer Manager](#) oder [CloudFront Vorsignieren](#) verwendet, lesen Sie den Abschnitt unter [the section called “Was unterscheidet sich zwischen 1.x und 2.x?”](#) Inhaltsverzeichnis.

Der hier beschriebene Ansatz ist ein Vorschlag. Sie können andere Techniken verwenden und die Codebearbeitungsfunktionen Ihrer IDE nutzen, um dasselbe Ergebnis zu erzielen.

## Übersicht über die Schritte

### 1. Fügen Sie zunächst die SDK for Java 2.x BOM hinzu

Indem Sie das Maven BOM (Bill of Material)-Element für das SDK for Java 2.x zu Ihrer POM-Datei hinzufügen, stellen Sie sicher, dass die gesamte benötigte v2-Abhängigkeit von derselben Version stammt. Ihr POM kann sowohl v1- als auch v2-Abhängigkeiten enthalten. Auf diese Weise können Sie Ihren Code schrittweise migrieren, anstatt ihn auf einmal zu ändern.

#### SDK für Java 2.x BOM

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.24.3</version>
      <type>pom</type>
```



```
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
```

Die [neueste Version](#) finden Sie im Maven Central Repository.

## 2. Suchen von Dateien nach Importanweisungen für v1-Klassen

Wenn Sie die Dateien in Ihrer Anwendung auf v1-Importe scannen, finden Sie v2-Abhängigkeiten, die Sie Ihrer Maven-POM-Datei hinzufügen können.

## 3. Ermitteln der v2-Maven-Abhängigkeiten anhand der v1-Importanweisungen

Nachdem Sie alle eindeutigen v1-Importanweisungen gefunden haben, können Sie das entsprechende Maven-Artefakt für die v2-Abhängigkeit ermitteln, indem Sie auf den Paketnamen auf die Abhängigkeitszuordnungstabelle verweisen.

## 4. Hinzufügen von v2-Abhängigkeitselementen zur POM-Datei

Aktualisieren Sie die Maven-POM-Datei mit Abhängigkeitselementen, die in Schritt 3 festgelegt wurden.

## 5. Ändern Sie in den Java-Dateien inkrementell über die v1-Klassen zu v2-Klassen

Nehmen Sie dabei die erforderlichen Änderungen vor, um die v2-API zu unterstützen, z. B. die Verwendung von Buildern anstelle von Konstruktoren und die Verwendung von fluent getters und setters.

## 6. Entfernen von v1-Maven-Abhängigkeiten aus dem POM und v1-Importen aus Dateien

Nehmen Sie dabei die erforderlichen Änderungen vor, um die v2-API zu unterstützen, z. B. die Verwendung von Buildern anstelle von Konstruktoren und die Verwendung der flüssigsten Getter und Setter.

## 7. Faktorwechsel des Codes zur Verwendung von v2-API-Verbesserungen

Nachdem der Code erfolgreich kompiliert wurde, während Tests bestanden hat, können Sie die Vorteile von v2-Verbesserungen nutzen, z. B. die Verwendung eines anderen HTTP-Clients oder anderer Paginatoren, um den Code zu vereinfachen. Dieser Schritt ist optional.

## Beispielmigration

In diesem Beispiel migrieren wir eine Anwendung, die das SDK für Java v1 verwendet und auf mehrere zugreift AWS-Services. In Schritt 5 gehen wir die folgende v1-Methode detailliert durch. Dies ist eine Methode in einer Klasse, die acht Methoden enthält und in der Anwendung 32 Klassen vorhanden sind.

### v1-Methode für die Migration

Nur die v1-SDK-Importe sind unten aus der Java-Datei aufgeführt.

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.RegionUtils;
import com.amazonaws.services.ec2.AmazonEC2Client;
import com.amazonaws.services.ec2.model.AmazonEC2Exception;
import com.amazonaws.services.ec2.model.CreateTagsRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesResult;
import com.amazonaws.services.ec2.model.Instance;
import com.amazonaws.services.ec2.model.InstanceStateName;
import com.amazonaws.services.ec2.model.Reservation;
import com.amazonaws.services.ec2.model.Tag;
import com.amazonaws.services.ec2.model.TerminateInstancesRequest;
...
private static List<Instance> getRunningInstances(AmazonEC2Client ec2, List<String>
instanceIds) {
    List<Instance> runningInstances = new ArrayList<>();
    try {
        DescribeInstancesRequest request = new DescribeInstancesRequest()
            .withInstanceIds(instanceIds);
        DescribeInstancesResult result;
        do {
            // DescribeInstancesResponse is a paginated response, so use tokens with
multiple requests.
            result = ec2.describeInstances(request);
            request.setNextToken(result.getNextToken()); // Prepare request for next
page.
            for (final Reservation r : result.getReservations()) {
                for (final Instance instance : r.getInstances()) {
                    LOGGER.info("Examining instanceId: " + instance.getInstanceId());
                    // if instance is in a running state, add it to runningInstances
list.

```

```

                if (RUNNING_STATES.contains(instance.getState().getName())) {
                    runningInstances.add(instance);
                }
            }
        } while (result.getNextToken() != null);
    } catch (final AmazonEC2Exception exception) {
        // if instance isn't found, assume its terminated and continue.
        if (exception.getErrorCode().equals(NOT_FOUND_ERROR_CODE)) {
            LOGGER.info("Instance probably terminated; moving on.");
        } else {
            throw exception;
        }
    }
    return runningInstances;
}

```

## 1. Hinzufügen von v2 Maven BOM

Fügen Sie die Maven BOM für das SDK for Java 2.x neben allen anderen Abhängigkeiten im `dependencyManagement` Abschnitt zur POM hinzu. Wenn Ihre POM-Datei über die BOM für v1 des SDK verfügt, belassen Sie sie vorerst. Sie wird in einem späteren Schritt entfernt.

### POM-Abhängigkeitsmanagement bei Beginn

```

<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.example</groupId>                <!--Existing dependency in POM. -->
      <artifactId>bom</artifactId>
      <version>1.3.4</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    ...
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-bom</artifactId>    <!--Existing v1 BOM dependency. -->
      <version>1.11.1000</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    ...

```

```
<dependency>
  <groupId>software.amazon.awssdk</groupId> <!--Add v2 BOM dependency. -->
  <artifactId>bom</artifactId>
  <version>2.24.3</version>
  <type>pom</type>
  <scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
```

## 2. Suchen von Dateien nach Importanweisungen für v1-Klassen

Suchen Sie im Code der Anwendung nach eindeutigen Vorkommen von `import com.amazonaws.services`. Dies hilft uns dabei, die v1-Abhängigkeiten zu ermitteln, die vom Projekt verwendet werden. Wenn Ihre Anwendung über eine Maven-POM-Datei mit aufgelisteten v1-Abhängigkeiten verfügt, können Sie stattdessen diese Informationen verwenden.

In diesem Beispiel verwenden wir den Befehl [ripgrep \(rg\)](#), um die Codebasis zu durchsuchen.

Führen Sie im Stammverzeichnis Ihrer Codebasis den folgenden `ripgrep` Befehl aus. Nachdem die Importanweisungen `ripgrep` gefunden hat, werden sie an die `uniq` Befehle `cut`, und weitergeleitet, um die Servicenamen zu isolieren.

```
rg --no-filename 'import\s+com\.amazonaws\.services' | cut -d '.' -f 4 | sort | uniq
```

Für diese Anwendung wird Folgendes in der -Konsole protokolliert.

```
autoscaling
cloudformation
ec2
identitymanagement
```

Dies weist darauf hin, dass mindestens ein Vorkommen jedes der folgenden Paketnamen in `import` Anweisungen verwendet wurde. Für unsere Zwecke spielen die einzelnen Klassennamen keine Rolle. Wir müssen nur die verwendeten Services finden.

```
com.amazonaws.services.autoscaling.*
com.amazonaws.services.cloudformation.*
com.amazonaws.services.ec2.*
com.amazonaws.services.identitymanagement.*
```

### 3. Ermitteln der v2-Maven-Abhängigkeiten anhand der v1-Importanweisungen

Die Servicenamen für v1, die wir von Schritt 2 isoliert haben, z. B. `autoscaling` und `cloudformation`, können größtenteils demselben v2-Servicenamen zugeordnet werden. Da die Maven-artifactId v2 in den meisten Fällen mit dem Servicenamen übereinstimmt, verfügen Sie über die Informationen, die Sie zum Hinzufügen von Abhängigkeitsblöcken zu Ihrer POM-Datei benötigen.

Die folgende Tabelle zeigt, wie wir die v2-Abhängigkeiten ermitteln können.

Der v1-Servicename ist ... Paketname	v2-Servicename ist ... Paketname	v2-Maven-Abhängigkeit
ec2  com.amazonaws.services. <b>ec2</b> .*	ec2  software.amazon.awssdk.services. <b>ec2</b> .*	<pre>&lt;dependency&gt;   &lt;groupId&gt;software. amazon.awssdk&lt;/gro upId&gt;   &lt;artifactId&gt; <b>ec2</b>&lt;/ artifactId&gt; &lt;/dependency&gt;</pre>
Auto Scaling  com.amazonaws.services. <b>autoscaling</b> .*	Auto Scaling  software.amazon.awssdk.services. <b>autoscaling</b> .*	<pre>&lt;dependency&gt;   &lt;groupId&gt;software. amazon.awssdk&lt;/gro upId&gt;   &lt;artifactId&gt; <b>autoscali ng</b> &lt;/artifactId&gt; &lt;/dependency&gt;</pre>
cloudformation  com.amazonaws.services. <b>cloudform ation</b> .*	cloudformation  software.amazon.awssdk. <b>cloudform ation</b> .*	<pre>&lt;dependency&gt;   &lt;groupId&gt;software. amazon.awssdk&lt;/gro upId&gt;   &lt;artifactId&gt; <b>cloudform ation</b> &lt;/artifactId&gt; &lt;/dependency&gt;</pre>
Identity Management*	iam*	<pre>&lt;dependency&gt;</pre>

Der v1-Servicename ist ...	v2-Servicename ist ...	v2-Maven-Abhängigkeit
Paketname	Paketname	
<code>com.amazonaws.services.<b>identitymanagement</b>.*</code>	<code>software.amazon.awssdk.<b>iam</b>.*</code>	<pre> &lt;groupId&gt;software. amazon.awssdk&lt;/gro upId&gt;   &lt;artifactId&gt; <b>iam</b>&lt;/ artifactId&gt; &lt;/dependency&gt; </pre>

\* Die iam Zuordnung von `identitymanagement` zu ist eine Ausnahme, bei der sich der im Paketname verwendete Servicename zwischen den Versionen unterscheidet.

#### 4. Hinzufügen von v2-Abhängigkeitselementen zur POM-Datei

In Schritt 3 haben wir die vier Abhängigkeitsblöcke bestimmt, die der POM-Datei hinzugefügt werden müssen. Wir müssen keine Version hinzufügen, da wir die BOM in Schritt 1 angegeben haben. Nachdem die Importe hinzugefügt wurden, hat unsere POM-Datei die folgenden Abhängigkeitselemente.

```

...
<dependencies>
  ...
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>autoscaling</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>iam</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>cloudformation</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>ec2</artifactId>
  </dependency>
  ...

```

```
</dependencies>
```

```
...
```

## 5. Ändern Sie in den Java-Dateien inkrementell über die v1-Klassen zu v2-Klassen

In der Methode, die wir migrieren, sehen wir

- Ein EC2-Service-Client von `com.amazonaws.services.ec2.AmazonEC2Client`.
- Mehrere verwendete EC2-Modellklassen. Zum Beispiel `DescribeInstancesRequest` und `DescribeInstancesResult`.

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.RegionUtils;
import com.amazonaws.services.ec2.AmazonEC2Client;
import com.amazonaws.services.ec2.model.AmazonEC2Exception;
import com.amazonaws.services.ec2.model.CreateTagsRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesResult;
import com.amazonaws.services.ec2.model.Instance;
import com.amazonaws.services.ec2.model.InstanceStateName;
import com.amazonaws.services.ec2.model.Reservation;
import com.amazonaws.services.ec2.model.Tag;
import com.amazonaws.services.ec2.model.TerminateInstancesRequest;
...
private static List<Instance> getRunningInstances(AmazonEC2Client ec2, List<String>
instanceIds)
    List<Instance> runningInstances = new ArrayList<>();
    try {
        DescribeInstancesRequest request = new DescribeInstancesRequest()
            .withInstanceIds(instanceIds);
        DescribeInstancesResult result;
        do {
            // DescribeInstancesResponse is a paginated response, so use tokens with
multiple re
            result = ec2.describeInstances(request);
            request.setNextToken(result.getNextToken()); // Prepare request for next
page.
            for (final Reservation r : result.getReservations()) {
                for (final Instance instance : r.getInstances()) {
                    LOGGER.info("Examining instanceId: " + instance.getInstanceId());
```

```

        // if instance is in a running state, add it to runningInstances
list.
        if (RUNNING_STATES.contains(instance.getState().getName())) {
            runningInstances.add(instance);
        }
    }
} while (result.getNextToken() != null);
} catch (final AmazonEC2Exception exception) {
    // if instance isn't found, assume its terminated and continue.
    if (exception.getErrorCode().equals(NOT_FOUND_ERROR_CODE)) {
        LOGGER.info("Instance probably terminated; moving on.");
    } else {
        throw exception;
    }
}
return runningInstances;
}
...

```

Unser Ziel ist es, alle v1-Importe durch v2-Importe zu ersetzen. Wir fahren jeweils eine Klasse fort.

#### a. Ersetzen der Importanweisung oder des Klassennamens

Wir sehen, dass der erste Parameter der `describeRunningInstances` Methode eine `v1AmazonEC2Client`-Instance ist. Führen Sie eine der folgenden Aktionen aus:

- Ersetzen Sie den Import für `com.amazonaws.services.ec2.AmazonEC2Client` durch `software.amazon.awssdk.services.ec2.Ec2Client` und ändern Sie `AmazonEC2Client` zu `Ec2Client`.
- Ändern Sie den Parametertyp in `Ec2Client` und lassen Sie die IDE uns zum richtigen Import auffordern. Unsere IDE wird uns auffordern, die v2-Klasse zu importieren, da sich die Clientnamen unterscheiden –`AmazonEC2Client` und `Ec2Client`. Dieser Ansatz funktioniert nicht, wenn der Klassenname in beiden Versionen gleich ist.

#### b. Ersetzen Sie v1-Modellklassen durch v2-Entsprechungen

`Ec2Client` Wenn wir nach der Änderung an v2 eine IDE verwenden, werden in der folgenden Anweisung Kompilierungsfehler angezeigt.

```
result = ec2.describeInstances(request);
```



Der Kompilierungsfehler ergibt sich aus der Verwendung einer Instance von v1 `DescribeInstancesRequest` als Parameter für die `v2-Ec2ClientdescribeInstances` Methode. Um das Problem zu beheben, machen Sie die folgenden Ersatz- oder Importanweisungen.

replace	mit
<pre>import com.amazonaws.services.ec2.model.DescribeInstancesRequest</pre>	<pre>import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest</pre>

c. Ändern Sie v1-Konstrukturen in v2 Builder.

Es treten immer noch Kompilierungsfehler auf, da es in den [v2-Klassen keine Konstruktoren](#) gibt. Nehmen Sie zur Behebung die folgende Änderung vor.

ändern	to
<pre>final DescribeInstancesRequest request = new DescribeInstancesRequest().withInstanceIds(instanceIdsCopy);</pre>	<pre>final DescribeInstancesRequest request = DescribeInstancesRequest.builder().instanceIds(instanceIdsCopy).build();</pre>

d. Ersetzen von v1-**\*Result**Antwortobjekten durch v2-**\*Response**Entsprechungen

Ein konsistenter Unterschied zwischen v1 und v2 besteht darin, dass alle [Antwortobjekte in v2 mit \\*Response anstelle von enden\\*Result](#). Ersetzen Sie den v1-`DescribeInstancesResult` Import in den v2-Import, `DescribeInstancesResponse`.

d. Nehmen Sie API-Änderungen vor

Die folgende Anweisung erfordert einige Änderungen.

```
request.setNextToken(result.getNextToken());
```

In v2 verwenden [Setter-Methoden](#) weder noch `set` mit `prefix`. Getter-Methoden mit dem Präfix `get` sind auch im SDK for Java 2.x weg

Modellklassen, wie z. B. die `request` -Instance, sind in v2 unveränderlich, daher müssen wir einen neuen `DescribeInstancesRequest` mit einem Builder erstellen.

In v2 wird die Anweisung wie folgt erstellt.

```
request = DescribeInstancesRequest.builder()
    .nextToken(result.nextToken())
    .build();
```

d. Wiederholen, bis die Methode mit v2-Klassen kompiliert wird

Fahren Sie mit dem Rest des Codes fort. Ersetzen Sie v1-Importe durch v2-Importe und beheben Sie die Kompilierungsfehler. Weitere Informationen finden Sie in der [v2-API-Referenz](#) und [Was ist eine andere Referenz](#) bei Bedarf.

Nachdem wir diese einzelne Methode migriert haben, haben wir den folgenden v2-Code.

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.RegionUtils;
import com.amazonaws.services.ec2.AmazonEC2Client;
import com.amazonaws.services.ec2.model.AmazonEC2Exception;
import com.amazonaws.services.ec2.model.CreateTagsRequest;
import com.amazonaws.services.ec2.model.InstanceStateName;
import com.amazonaws.services.ec2.model.Tag;
import com.amazonaws.services.ec2.model.TerminateInstancesRequest;

import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.Instance;
import software.amazon.awssdk.services.ec2.model.Reservation;
...
private static List<Instance> getRunningInstances(Ec2Client ec2, List<String>
instanceIds) {
    List<Instance> runningInstances = new ArrayList<>();
    try {
        DescribeInstancesRequest request = DescribeInstancesRequest.builder()
            .instanceIds(instanceIds)
```

```

        .build();
    DescribeInstancesResponse result;
    do {
        // DescribeInstancesResponse is a paginated response, so use tokens
with multiple re
        result = ec2.describeInstances(request);
        request = DescribeInstancesRequest.builder() // Prepare request for
next page.
            .nextToken(result.nextToken())
            .build();
        for (final Reservation r : result.reservations()) {
            for (final Instance instance : r.instances()) {
                // if instance is in a running state, add it to
runningInstances list.
                if (RUNNING_STATES.contains(instance.state().nameAsString())) {
                    runningInstances.add(instance);
                }
            }
        }
    } while (result.nextToken() != null);
} catch (final Ec2Exception exception) {
    // if instance isn't found, assume its terminated and continue.
    if (exception.awsErrorDetails().errorCode().equals(NOT_FOUND_ERROR_CODE)) {
        LOGGER.info("Instance probably terminated; moving on.");
    } else {
        throw exception;
    }
}
return runningInstances;
}
...

```

Da wir eine einzelne Methode in einer Java-Datei mit acht Methoden migrieren, haben wir eine Mischung aus v1- und v2-Importen, während wir die Datei durcharbeiten. Wir haben die letzten sechs Importanweisungen hinzugefügt, während wir die Schritte ausgeführt haben.

Nachdem wir den gesamten Code migriert haben, gibt es keine v1-Importanweisungen mehr.

## 6. Entfernen von v1-Maven-Abhängigkeiten aus dem POM und v1-Importen aus Dateien

Nachdem wir den gesamten v1-Code in der Datei migriert haben, haben wir die folgenden v2-SDK-Importanweisungen.

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.regions.ServiceMetadata;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.Instance;
import software.amazon.awssdk.services.ec2.model.InstanceStateName;
import software.amazon.awssdk.services.ec2.model.Reservation;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesRequest;
```

Nachdem wir alle Dateien in unserer Anwendung migriert haben, benötigen wir die v1-Abhängigkeiten in unserer POM-Datei nicht mehr. Entfernen Sie die v1-BOM aus dem Abschnitt `dependencyManagement`, falls verwendet, und alle v1-Abhängigkeitsblöcke.

## 7. Faktorwechsel des Codes zur Verwendung von v2-API-Verbesserungen

Für den Ausschnitt, den wir migriert haben, können wir optional einen v2-Paginator verwenden und das SDK die tokenbasierten Anfragen für weitere Daten verwalten lassen.

Wir können die gesamte `-doKlausel` durch Folgendes ersetzen.

```
DescribeInstancesIterable responses =
ec2.describeInstancesPaginator(request);

responses.reservations().stream()
    .forEach(reservation -> reservation.instances()
        .forEach(instance -> {
            if
(RUNNING_STATES.contains(instance.state().nameAsString())) {
                runningInstances.put(instance.instanceId(),
instance);
            }
        }));
```

# Was unterscheidet sich zwischen AWS SDK for Java 1.x und 2.x?

In diesem Abschnitt werden die wichtigsten Änderungen beschrieben, die Sie beachten sollten, wenn Sie eine Anwendung von der AWS SDK for Java Version 1.x in die Version 2.x konvertieren.

## Änderung des Paketnamens

Eine spürbare Änderung vom SDK für Java 1.x zum SDK für Java 2.x ist die Änderung des Paketnamens. Paketnamen beginnen mit `software.amazon.awssdk` in SDK 2.x, wohingegen das SDK 1.x verwendet `com.amazonaws`.

Dieselben Namen unterscheiden Maven-Artefakte von SDK 1.x zu SDK 2.x. Maven-Artefakte für das SDK 2.x verwenden die `software.amazon.awssdk` groupId, während das SDK 1.x die `com.amazonaws` groupId verwendet.

Es gibt einige Male, wenn Ihr Code eine `com.amazonaws` Abhängigkeit für ein Projekt erfordert, das andernfalls nur SDK-2.x-Artefakte verwendet. Ein Beispiel dafür ist, wenn Sie mit serverseitig arbeiten AWS Lambda. Dies wurde im Abschnitt [Einrichten eines Apache-Maven-Projekts](#) weiter oben in diesem Leitfaden gezeigt.

### Note

Mehrere Paketnamen im SDK 1.x enthalten v2. Die Verwendung von v2 in diesem Fall bedeutet in der Regel, dass Code im Paket auf die Arbeit mit Version 2 des Services ausgerichtet ist.

Da der vollständige Paketname mit `com.amazonaws` beginnt, handelt es sich um SDK-1.x-Komponenten. Beispiele für diese Paketnamen im SDK 1.x sind:

- `com.amazonaws.services.dynamodbv2`
- `com.amazonaws.retry.v2`
- `com.amazonaws.services.apigatewayv2`
- `com.amazonaws.services.simpleemailv2`

## Hinzufügen von Version 2.x zu Ihrem Projekt

Maven ist die empfohlene Methode zur Verwaltung von Abhängigkeiten bei Verwendung von AWS SDK for Java 2.x. Um Version 2.x-Komponenten zu Ihrem Projekt hinzuzufügen, aktualisieren Sie Ihre `pom.xml` Datei mit einer Abhängigkeit vom SDK.

### Example

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.16.1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>dynamodb</artifactId>
  </dependency>
</dependencies>
```

Sie können auch [Version 1.x und 2.x verwenden side-by-side](#), wenn Sie Ihr Projekt auf Version 2.x migrieren.

## Unveränderliche POJOs

Clients und Operationsanforderungs- und Antwortobjekte sind jetzt unveränderlich und können nach der Erstellung nicht geändert werden. Zur Wiederverwendung einer Anforderungs- oder Antwortvariable müssen Sie ein neues Objekt erstellen, das sie ihr zuweisen können.

### Example Aktualisieren eines Anfrageobjekts in 1.x

```
DescribeAlarmsRequest request = new DescribeAlarmsRequest();
DescribeAlarmsResult response = cw.describeAlarms(request);

request.setNextToken(response.getNextToken());
```

## Example Aktualisieren eines Anfrageobjekts in 2.x

```
DescribeAlarmsRequest request = DescribeAlarmsRequest.builder().build();
DescribeAlarmsResponse response = cw.describeAlarms(request);

request = DescribeAlarmsRequest.builder()
    .nextToken(response.nextToken())
    .build();
```

## Setter- und Getter-Methoden

Im AWS SDK for Java 2.x enthalten die Namen der Setter-Methode das `with` Präfix `set` oder nicht. Zum Beispiel `*.withEndpoint()` ist jetzt `*.endpoint()`.

Getter-Methodennamen verwenden das `get` Präfix nicht.

Example für die Verwendung von Setter-Methoden in 1.x

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
    .withRegion("us-east-1")
    .build();
```

Example für die Verwendung von Setter-Methoden in 2.x

```
DynamoDbClient client = DynamoDbClient.builder()
    .region(Region.US_EAST_1)
    .build();
```

Example für die Verwendung von Getter-Methoden in 1.x

```
String token = request.getNextToken();
```

Example für die Verwendung von Getter-Methoden in 2.x

```
String token = request.nextToken();
```

## Modellklassennamen

Modellklassennamen, die Serviceantworten darstellen, enden mit `Response` in v2, anstatt `Result` dass v1 verwendet.

## Example Klassennamen, die eine Antwort in v1 darstellen

```
CreateApiKeyResult
AllocateAddressResult
```

## Example Klassennamen, die eine Antwort in v2 darstellen

```
CreateApiKeyResponse
AllocateAddressResponse
```

## Migrationsstatus von Bibliotheken und Dienstprogrammen

### Bibliotheken und Dienstprogramme von SDK für Java

In der folgenden Tabelle ist der Migrationsstatus von Bibliotheken und Dienstprogrammen für das SDK for Java aufgeführt.

Version 1.12.x Name	Version 2.x Name	Seit -Version in 2.x
DynamoDBMapper	<a href="#">DynamoDbEnhancedClient</a>	2.12.0
Waiter	<a href="#">Waiter</a>	2.15.0
CloudFrontUrlSigner, CloudFrontCookieSigner	<a href="#">CloudFrontUtilities</a>	2.18.33
TransferManager	<a href="#">S3TransferManager</a>	2.19.0
EC2-Metadaten-Client	<a href="#">EC2-Metadaten-Client</a>	2.19.29
S3-URI-Parser	<a href="#">S3-URI-Parser</a>	2.20.41
IAM-Richtlinien-Builder	<a href="#">IAM-Richtlinien-Builder</a>	2.20.126
Clientseitige Pufferung in Amazon SQS	Automatisches Anforderungs- Batching	<a href="#">noch nicht veröffentlicht</a>
Fortschritts-Listener	Fortschritts-Listener	<a href="#">noch nicht veröffentlicht</a>



## Verwandte Bibliotheken

In der folgenden Tabelle sind Bibliotheken aufgeführt, die separat veröffentlicht werden, aber mit dem SDK for Java 2.x arbeiten.

Name, der mit Version 2.x des SDK for Java verwendet wird	Seit Version
<a href="#">Amazon S3-Verschlüsselungsclient</a>	3.0.0 <sup>1</sup>
<a href="#">AWS Datenbankverschlüsselungs-Client für DynamoDB</a>	3.0.0 <sup>2</sup>

<sup>1</sup>Der Verschlüsselungsclient für Amazon S3 ist mithilfe der folgenden Maven-Abhängigkeit verfügbar.

```
<dependency>
  <groupId>software.amazon.encryption.s3</groupId>
  <artifactId>amazon-s3-encryption-client-java</artifactId>
  <version>3.x</version>
</dependency>
```

<sup>2</sup>Der AWS Database Encryption Client für DynamoDB ist verfügbar, indem die folgende Maven-Abhängigkeit verwendet wird.

```
<dependency>
  <groupId>software.amazon.cryptography</groupId>
  <artifactId>aws-database-encryption-sdk-dynamodb</artifactId>
  <version>3.x</version>
</dependency>
```

## Migrationsdetails für Bibliotheken und Dienstprogramme

- [S3 Transfer Manager](#)
- [EC2-Metadaten-Hilfsprogramm](#)
- [CloudFront Vorsignieren](#)
- [S3-URI-Parsing](#)

# Client-Änderungen

## Client Builder

Sie müssen alle Clients mit der Client-Builder-Methode erstellen. Konstruktoren sind nicht mehr verfügbar.

### Example Erstellen eines Clients in Version 1.x

```
AmazonDynamoDB ddbClient = AmazonDynamoDBClientBuilder.defaultClient();
AmazonDynamoDBClient ddbClient = new AmazonDynamoDBClient();
```

### Example Erstellen eines Clients in Version 2.x

```
DynamoDbClient ddbClient = DynamoDbClient.create();
DynamoDbClient ddbClient = DynamoDbClient.builder().build();
```

## Client-Klassennamen

Alle Namen der Client-Klassen sind jetzt vollständig mit Groß- und Kleinschreibung versehen und haben kein Präfix mehr von Amazon. Diese Änderungen sind entsprechend der in der AWS CLI verwendeten Namen ausgerichtet.

### Example Klassennamen in 1.x

```
AmazonDynamoDB
AWSACMPAAsyncClient
```

### Example Klassennamen in 2.x

```
DynamoDbClient
AcmAsyncClient
```

## Änderungen des Client-Klassennamens

1.x-Client	2.x-Client
com.amazonaws.services.acmpca.AWSACMPAAsyncClient	software.amazon.awssdk.services.acm.AcmAsyncClient

1.x-Client	2.x-Client
<code>com.amazonaws.services.acmpca.AWSACMPCAClient</code>	<code>software.amazon.awssdk.services.acm.AcmClient</code>
<code>com.amazonaws.services.alexforbusiness.AmazonAlexaForBusinessAsyncClient</code>	<code>software.amazon.awssdk.services.alexforbusiness.AlexaForBusinessAsyncClient</code>
<code>com.amazonaws.services.alexforbusiness.AmazonAlexaForBusinessClient</code>	<code>software.amazon.awssdk.services.alexforbusiness.AlexaForBusinessClient</code>
<code>com.amazonaws.services.apigateway.AmazonApiGatewayAsyncClient</code>	<code>software.amazon.awssdk.services.apigateway.ApiGatewayAsyncClient</code>
<code>com.amazonaws.services.apigateway.AmazonApiGatewayClient</code>	<code>software.amazon.awssdk.services.apigateway.ApiGatewayClient</code>
<code>com.amazonaws.services.applicationautoscaling.AWSApplicationAutoScalingAsyncClient</code>	<code>software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingAsyncClient</code>
<code>com.amazonaws.services.applicationautoscaling.AWSApplicationAutoScalingClient</code>	<code>software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient</code>
<code>com.amazonaws.services.applicationdiscovery.AWSApplicationDiscoveryAsyncClient</code>	<code>software.amazon.awssdk.services.applicationdiscovery.ApplicationDiscoveryAsyncClient</code>
<code>com.amazonaws.services.applicationdiscovery.AWSApplicationDiscoveryClient</code>	<code>software.amazon.awssdk.services.applicationdiscovery.ApplicationDiscoveryClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.appstream.AmazonAppStreamAsyncClient</code>	<code>software.amazon.awssdk.services.appstream.AppStreamAsyncClient</code>
<code>com.amazonaws.services.appstream.AmazonAppStreamClient</code>	<code>software.amazon.awssdk.services.appstream.AppStreamClient</code>
<code>com.amazonaws.services.appsync.AWSAppSyncAsyncClient</code>	<code>software.amazon.awssdk.services.appsync.AppSyncAsyncClient</code>
<code>com.amazonaws.services.appsync.AWSAppSyncClient</code>	<code>software.amazon.awssdk.services.appsync.AppSyncClient</code>
<code>com.amazonaws.services.athena.AmazonAthenaAsyncClient</code>	<code>software.amazon.awssdk.services.athena.AthenaAsyncClient</code>
<code>com.amazonaws.services.athena.AmazonAthenaClient</code>	<code>software.amazon.awssdk.services.athena.AthenaClient</code>
<code>com.amazonaws.services.autoscaling.AmazonAutoScalingAsyncClient</code>	<code>software.amazon.awssdk.services.autoscaling.AutoScalingAsyncClient</code>
<code>com.amazonaws.services.autoscaling.AmazonAutoScalingClient</code>	<code>software.amazon.awssdk.services.autoscaling.AutoScalingClient</code>
<code>com.amazonaws.services.autoscalingplans.AWSAutoScalingPlansAsyncClient</code>	<code>software.amazon.awssdk.services.autoscalingplans.AutoScalingPlansAsyncClient</code>
<code>com.amazonaws.services.autoscalingplans.AWSAutoScalingPlansClient</code>	<code>software.amazon.awssdk.services.autoscalingplans.AutoScalingPlansClient</code>
<code>com.amazonaws.services.batch.AWSBatchAsyncClient</code>	<code>software.amazon.awssdk.services.batch.BatchAsyncClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.batch.AWSBatchClient</code>	<code>software.amazon.awssdk.services.batch.BatchClient</code>
<code>com.amazonaws.services.budgets.AWSBudgetsAsyncClient</code>	<code>software.amazon.awssdk.services.budgets.BudgetsAsyncClient</code>
<code>com.amazonaws.services.budgets.AWSBudgetsClient</code>	<code>software.amazon.awssdk.services.budgets.BudgetsClient</code>
<code>com.amazonaws.services.certificatemanager.AWSCertificateManagerAsyncClient</code>	<code>software.amazon.awssdk.services.acm.AcmAsyncClient</code>
<code>com.amazonaws.services.certificatemanager.AWSCertificateManagerClient</code>	<code>software.amazon.awssdk.services.acm.AcmClient</code>
<code>com.amazonaws.services.cloud9.AWSCloud9AsyncClient</code>	<code>software.amazon.awssdk.services.cloud9.Cloud9AsyncClient</code>
<code>com.amazonaws.services.cloud9.AWSCloud9Client</code>	<code>software.amazon.awssdk.services.cloud9.Cloud9Client</code>
<code>com.amazonaws.services.clouddirectory.AmazonCloudDirectoryAsyncClient</code>	<code>software.amazon.awssdk.services.clouddirectory.CloudDirectoryAsyncClient</code>
<code>com.amazonaws.services.clouddirectory.AmazonCloudDirectoryClient</code>	<code>software.amazon.awssdk.services.clouddirectory.CloudDirectoryClient</code>
<code>com.amazonaws.services.cloudformation.AmazonCloudFormationAsyncClient</code>	<code>software.amazon.awssdk.services.cloudformation.CloudFormationAsyncClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.cloudformation.AmazonCloudFormationClient</code>	<code>software.amazon.awssdk.services.cloudformation.CloudFormationClient</code>
<code>com.amazonaws.services.cloudfront.AmazonCloudFrontAsyncClient</code>	<code>software.amazon.awssdk.services.cloudfront.CloudFrontAsyncClient</code>
<code>com.amazonaws.services.cloudfront.AmazonCloudFrontClient</code>	<code>software.amazon.awssdk.services.cloudfront.CloudFrontClient</code>
<code>com.amazonaws.services.cloudhsm.AWSCloudHSMAsyncClient</code>	<code>software.amazon.awssdk.services.cloudhsm.CloudHsmAsyncClient</code>
<code>com.amazonaws.services.cloudhsm.AWSCloudHSMClient</code>	<code>software.amazon.awssdk.services.cloudhsm.CloudHsmClient</code>
<code>com.amazonaws.services.cloudhsmv2.AWSCloudHSMV2AsyncClient</code>	<code>software.amazon.awssdk.services.cloudhsmv2.CloudHsmV2AsyncClient</code>
<code>com.amazonaws.services.cloudhsmv2.AWSCloudHSMV2Client</code>	<code>software.amazon.awssdk.services.cloudhsmv2.CloudHsmV2Client</code>
<code>com.amazonaws.services.cloudsearchdomain.AmazonCloudSearchDomainAsyncClient</code>	<code>software.amazon.awssdk.services.cloudsearchdomain.CloudSearchDomainAsyncClient</code>
<code>com.amazonaws.services.cloudsearchdomain.AmazonCloudSearchDomainClient</code>	<code>software.amazon.awssdk.services.cloudsearchdomain.CloudSearchDomainClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.cloudsearchv2.AmazonCloudSearchAsyncClient</code>	<code>software.amazon.awssdk.services.cloudsearch.CloudSearchAsyncClient</code>
<code>com.amazonaws.services.cloudsearchv2.AmazonCloudSearchClient</code>	<code>software.amazon.awssdk.services.cloudsearch.CloudSearchClient</code>
<code>com.amazonaws.services.cloudtrail.AWSCloudTrailAsyncClient</code>	<code>software.amazon.awssdk.services.cloudtrail.CloudTrailAsyncClient</code>
<code>com.amazonaws.services.cloudtrail.AWSCloudTrailClient</code>	<code>software.amazon.awssdk.services.cloudtrail.CloudTrailClient</code>
<code>com.amazonaws.services.cloudwatch.AmazonCloudWatchAsyncClient</code>	<code>software.amazon.awssdk.services.cloudwatch.CloudWatchAsyncClient</code>
<code>com.amazonaws.services.cloudwatch.AmazonCloudWatchClient</code>	<code>software.amazon.awssdk.services.cloudwatch.CloudWatchClient</code>
<code>com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsAsyncClient</code>	<code>software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsAsyncClient</code>
<code>com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsClient</code>	<code>software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient</code>
<code>com.amazonaws.services.codebuild.AWSCodeBuildAsyncClient</code>	<code>software.amazon.awssdk.services.codebuild.CodeBuildAsyncClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.codebuild.AWSCodeBuildClient</code>	<code>software.amazon.awssdk.services.codebuild.CodeBuildClient</code>
<code>com.amazonaws.services.codecommit.AWSCodeCommitAsyncClient</code>	<code>software.amazon.awssdk.services.codecommit.CodeCommitAsyncClient</code>
<code>com.amazonaws.services.codecommit.AWSCodeCommitClient</code>	<code>software.amazon.awssdk.services.codecommit.CodeCommitClient</code>
<code>com.amazonaws.services.codedeploy.AmazonCodeDeployAsyncClient</code>	<code>software.amazon.awssdk.services.codedeploy.CodeDeployAsyncClient</code>
<code>com.amazonaws.services.codedeploy.AmazonCodeDeployClient</code>	<code>software.amazon.awssdk.services.codedeploy.CodeDeployClient</code>
<code>com.amazonaws.services.codepipeline.AWSCodePipelineAsyncClient</code>	<code>software.amazon.awssdk.services.codepipeline.CodePipelineAsyncClient</code>
<code>com.amazonaws.services.codepipeline.AWSCodePipelineClient</code>	<code>software.amazon.awssdk.services.codepipeline.CodePipelineClient</code>
<code>com.amazonaws.services.codestar.AWSCodeStarAsyncClient</code>	<code>software.amazon.awssdk.services.codestar.CodeStarAsyncClient</code>
<code>com.amazonaws.services.codestar.AWSCodeStarClient</code>	<code>software.amazon.awssdk.services.codestar.CodeStarClient</code>
<code>com.amazonaws.services.cognitoidentity.AmazonCognitoIdentityAsyncClient</code>	<code>software.amazon.awssdk.services.cognitoidentity.CognitoIdentityAsyncClient</code>



1.x-Client	2.x-Client
<code>com.amazonaws.services.cognitoidentity.AmazonCognitoIdentityClient</code>	<code>software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient</code>
<code>com.amazonaws.services.cognitoidentityp.AWSCognitoIdentityProviderAsyncClient</code>	<code>software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderAsyncClient</code>
<code>com.amazonaws.services.cognitoidentityp.AWSCognitoIdentityProviderClient</code>	<code>software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient</code>
<code>com.amazonaws.services.cognitosync.AmazonCognitoSyncAsyncClient</code>	<code>software.amazon.awssdk.services.cognitosync.CognitoSyncAsyncClient</code>
<code>com.amazonaws.services.cognitosync.AmazonCognitoSyncClient</code>	<code>software.amazon.awssdk.services.cognitosync.CognitoSyncClient</code>
<code>com.amazonaws.services.comprehend.AmazonComprehendAsyncClient</code>	<code>software.amazon.awssdk.services.comprehend.ComprehendAsyncClient</code>
<code>com.amazonaws.services.comprehend.AmazonComprehendClient</code>	<code>software.amazon.awssdk.services.comprehend.ComprehendClient</code>
<code>com.amazonaws.services.config.AmazonConfigAsyncClient</code>	<code>software.amazon.awssdk.services.config.ConfigAsyncClient</code>
<code>com.amazonaws.services.config.AmazonConfigClient</code>	<code>software.amazon.awssdk.services.config.ConfigClient</code>
<code>com.amazonaws.services.connect.AmazonConnectAsyncClient</code>	<code>software.amazon.awssdk.services.connect.ConnectAsyncClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.connect.AmazonConnectClient</code>	<code>software.amazon.awssdk.services.connect.ConnectClient</code>
<code>com.amazonaws.services.costandusagereport.AWSCostAndUsageReportAsyncClient</code>	<code>software.amazon.awssdk.services.costandusagereport.CostAndUsageReportAsyncClient</code>
<code>com.amazonaws.services.costandusagereport.AWSCostAndUsageReportClient</code>	<code>software.amazon.awssdk.services.costandusagereport.CostAndUsageReportClient</code>
<code>com.amazonaws.services.costexplorer.AWSCostExplorerAsyncClient</code>	<code>software.amazon.awssdk.services.costexplorer.CostExplorerAsyncClient</code>
<code>com.amazonaws.services.costexplorer.AWSCostExplorerClient</code>	<code>software.amazon.awssdk.services.costexplorer.CostExplorerClient</code>
<code>com.amazonaws.services.databasemigrationservice.AWSDatabaseMigrationServiceAsyncClient</code>	<code>software.amazon.awssdk.services.databasemigration.DatabaseMigrationAsyncClient</code>
<code>com.amazonaws.services.databasemigrationservice.AWSDatabaseMigrationServiceClient</code>	<code>software.amazon.awssdk.services.databasemigration.DatabaseMigrationClient</code>
<code>com.amazonaws.services.datapipeline.DataPipelineAsyncClient</code>	<code>software.amazon.awssdk.services.datapipeline.DataPipelineAsyncClient</code>
<code>com.amazonaws.services.datapipeline.DataPipelineClient</code>	<code>software.amazon.awssdk.services.datapipeline.DataPipelineAsyncClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.dax. AmazonDaxAsyncClient</code>	<code>software.amazon.awssdk.serv ices.dax.DaxAsyncClient</code>
<code>com.amazonaws.services.dax. AmazonDaxClient</code>	<code>software.amazon.awssdk.serv ices.dax.DaxClient</code>
<code>com.amazonaws.services.devi cefarm.AWSDeviceFarmAsyncClient</code>	<code>software.amazon.awssdk.serv ices.devicefarm.DeviceFarmA syncClient</code>
<code>com.amazonaws.services.devi cefarm.AWSDeviceFarmClient</code>	<code>software.amazon.awssdk.serv ices.devicefarm.DeviceFarmC lient</code>
<code>com.amazonaws.services.dire ctconnect.AmazonDirectConne ctAsyncClient</code>	<code>software.amazon.awssdk.serv ices.directconnect.DirectCo nnectAsyncClient</code>
<code>com.amazonaws.services.dire ctconnect.AmazonDirectConne ctClient</code>	<code>software.amazon.awssdk.serv ices.directconnect.DirectCo nnectClient</code>
<code>com.amazonaws.services.dire ctory.AWSDirectoryServiceAs yncClient</code>	<code>software.amazon.awssdk.serv ices.directory.DirectoryAsy ncClient</code>
<code>com.amazonaws.services.dire ctory.AWSDirectoryServiceClient</code>	<code>software.amazon.awssdk.serv ices.directory.DirectoryClient</code>
<code>com.amazonaws.services.dlm. AmazonDLMAsyncClient</code>	<code>software.amazon.awssdk.serv ices.dlm.DlmAsyncClient</code>
<code>com.amazonaws.services.dlm. AmazonDLMClient</code>	<code>software.amazon.awssdk.serv ices.dlm.DlmClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBAsyncClient</code>	<code>software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBClient</code>	<code>software.amazon.awssdk.services.dynamodb.DynamoDbClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsAsyncClient</code>	<code>software.amazon.awssdk.services.dynamodb.streams.DynamoDbStreamsAsyncClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsClient</code>	<code>software.amazon.awssdk.services.dynamodb.streams.DynamoDbStreamsClient</code>
<code>com.amazonaws.services.ec2.AmazonEC2AsyncClient</code>	<code>software.amazon.awssdk.services.ec2.Ec2AsyncClient</code>
<code>com.amazonaws.services.ec2.AmazonEC2Client</code>	<code>software.amazon.awssdk.services.ec2.Ec2Client</code>
<code>com.amazonaws.services.ecr.AmazonECRAsyncClient</code>	<code>software.amazon.awssdk.services.ecr.EcrAsyncClient</code>
<code>com.amazonaws.services.ecr.AmazonECRClient</code>	<code>software.amazon.awssdk.services.ecr.EcrClient</code>
<code>com.amazonaws.services.ecs.AmazonECSAsyncClient</code>	<code>software.amazon.awssdk.services.ecs.EcsAsyncClient</code>
<code>com.amazonaws.services.ecs.AmazonECSClient</code>	<code>software.amazon.awssdk.services.ecs.EcsClient</code>
<code>com.amazonaws.services.eks.AmazonEKSAsyncClient</code>	<code>software.amazon.awssdk.services.eks.EksAsyncClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.eks. AmazonEKSClient</code>	<code>software.amazon.awssdk.serv ices.eks.EksClient</code>
<code>com.amazonaws.services.elas ticache.AmazonElasticCacheAs yncClient</code>	<code>software.amazon.awssdk.serv ices.elasticache.ElastiCach eAsyncClient</code>
<code>com.amazonaws.services.elas ticache.AmazonElasticCacheClient</code>	<code>software.amazon.awssdk.serv ices.elasticache.ElastiCach eClient</code>
<code>com.amazonaws.services.elas ticbeanstalk.AWSElasticBean stalkAsyncClient</code>	<code>software.amazon.awssdk.serv ices.elasticbeanstalk.Elast icBeanstalkAsyncClient</code>
<code>com.amazonaws.services.elas ticbeanstalk.AWSElasticBean stalkClient</code>	<code>software.amazon.awssdk.serv ices.elasticbeanstalk.Elast icBeanstalkClient</code>
<code>com.amazonaws.services.elas ticfilesystem.AmazonElastic FileSystemAsyncClient</code>	<code>software.amazon.awssdk.serv ices.efs.EfsAsyncClient</code>
<code>com.amazonaws.services.elas ticfilesystem.AmazonElastic FileSystemClient</code>	<code>software.amazon.awssdk.serv ices.efs.EfsClient</code>
<code>com.amazonaws.services.elas ticloadbalancing.AmazonElas ticLoadBalancingAsyncClient</code>	<code>software.amazon.awssdk.serv ices.elasticloadbalancing.E lasticLoadBalancingAsyncClient</code>
<code>com.amazonaws.services.elas ticloadbalancing.AmazonElas ticLoadBalancingClient</code>	<code>software.amazon.awssdk.serv ices.elasticloadbalancing.E lasticLoadBalancingClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.elasticloadbalancingv2.AmazonElasticLoadBalancingAsyncClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancingv2.ElasticLoadBalancingV2AsyncClient</code>
<code>com.amazonaws.services.elasticloadbalancingv2.AmazonElasticLoadBalancingClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancingv2.ElasticLoadBalancingV2Client</code>
<code>com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceAsyncClient</code>	<code>software.amazon.awssdk.services.emr.EmrAsyncClient</code>
<code>com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceClient</code>	<code>software.amazon.awssdk.services.emr.EmrClient</code>
<code>com.amazonaws.services.elasticsearch.AWSElasticsearchAsyncClient</code>	<code>software.amazon.awssdk.services.elasticsearch.ElasticsearchAsyncClient</code>
<code>com.amazonaws.services.elasticsearch.AWSElasticsearchClient</code>	<code>software.amazon.awssdk.services.elasticsearch.ElasticsearchClient</code>
<code>com.amazonaws.services.elastictranscoder.AmazonElasticTranscoderAsyncClient</code>	<code>software.amazon.awssdk.services.elastictranscoder.ElasticTranscoderAsyncClient</code>
<code>com.amazonaws.services.elastictranscoder.AmazonElasticTranscoderClient</code>	<code>software.amazon.awssdk.services.elastictranscoder.ElasticTranscoderClient</code>
<code>com.amazonaws.services.fms.AWSFMSAsyncClient</code>	<code>software.amazon.awssdk.services.fms.FmsAsyncClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.fms.AWSFMSClient</code>	<code>software.amazon.awssdk.services.fms.FmsClient</code>
<code>com.amazonaws.services.gamelift.AmazonGameLiftAsyncClient</code>	<code>software.amazon.awssdk.services.gamelift.GameLiftAsyncClient</code>
<code>com.amazonaws.services.gamelift.AmazonGameLiftClient</code>	<code>software.amazon.awssdk.services.gamelift.GameLiftClient</code>
<code>com.amazonaws.services.glacier.AmazonGlacierAsyncClient</code>	<code>software.amazon.awssdk.services.glacier.GlacierAsyncClient</code>
<code>com.amazonaws.services.glacier.AmazonGlacierClient</code>	<code>software.amazon.awssdk.services.glacier.GlacierClient</code>
<code>com.amazonaws.services.glue.AWSGlueAsyncClient</code>	<code>software.amazon.awssdk.services.glue.GlueAsyncClient</code>
<code>com.amazonaws.services.glue.AWSGlueClient</code>	<code>software.amazon.awssdk.services.glue.GlueClient</code>
<code>com.amazonaws.services.greengrass.AWSGreengrassAsyncClient</code>	<code>software.amazon.awssdk.services.greengrass.GreengrassAsyncClient</code>
<code>com.amazonaws.services.greengrass.AWSGreengrassClient</code>	<code>software.amazon.awssdk.services.greengrass.GreengrassClient</code>
<code>com.amazonaws.services.guardduty.AmazonGuardDutyAsyncClient</code>	<code>software.amazon.awssdk.services.guardduty.GuardDutyAsyncClient</code>
<code>com.amazonaws.services.guardduty.AmazonGuardDutyClient</code>	<code>software.amazon.awssdk.services.guardduty.GuardDutyClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.health.AWSHealthAsyncClient</code>	<code>software.amazon.awssdk.services.health.HealthAsyncClient</code>
<code>com.amazonaws.services.health.AWSHealthClient</code>	<code>software.amazon.awssdk.services.health.HealthClient</code>
<code>com.amazonaws.services.identitymanagement.AmazonIdentityManagementAsyncClient</code>	<code>software.amazon.awssdk.services.iam.IamAsyncClient</code>
<code>com.amazonaws.services.identitymanagement.AmazonIdentityManagementClient</code>	<code>software.amazon.awssdk.services.iam.IamClient</code>
<code>com.amazonaws.services.importexport.AmazonImportExportAsyncClient</code>	<code>software.amazon.awssdk.services.importexport.ImportExportAsyncClient</code>
<code>com.amazonaws.services.importexport.AmazonImportExportClient</code>	<code>software.amazon.awssdk.services.importexport.ImportExportClient</code>
<code>com.amazonaws.services.inspector.AmazonInspectorAsyncClient</code>	<code>software.amazon.awssdk.services.inspector.InspectorAsyncClient</code>
<code>com.amazonaws.services.inspector.AmazonInspectorClient</code>	<code>software.amazon.awssdk.services.inspector.InspectorClient</code>
<code>com.amazonaws.services.iot.AWSIoTAsyncClient</code>	<code>software.amazon.awssdk.services.iot.IotAsyncClient</code>
<code>com.amazonaws.services.iot.AWSIoTClient</code>	<code>software.amazon.awssdk.services.iot.IotClient</code>



1.x-Client	2.x-Client
<code>com.amazonaws.services.iot1clickdevices.AWSIoT1ClickDevicesAsyncClient</code>	<code>software.amazon.awssdk.services.iot1clickdevices.Iot1ClickDevicesAsyncClient</code>
<code>com.amazonaws.services.iot1clickdevices.AWSIoT1ClickDevicesClient</code>	<code>software.amazon.awssdk.services.iot1clickdevices.Iot1ClickDevicesClient</code>
<code>com.amazonaws.services.iot1clickprojects.AWSIoT1ClickProjectsAsyncClient</code>	<code>software.amazon.awssdk.services.iot1clickprojects.Iot1ClickProjectsAsyncClient</code>
<code>com.amazonaws.services.iot1clickprojects.AWSIoT1ClickProjectsClient</code>	<code>software.amazon.awssdk.services.iot1clickprojects.Iot1ClickProjectsClient</code>
<code>com.amazonaws.services.iotanalytics.AWSIoTAnalyticsAsyncClient</code>	<code>software.amazon.awssdk.services.iotanalytics.IotAnalyticsAsyncClient</code>
<code>com.amazonaws.services.iotanalytics.AWSIoTAnalyticsClient</code>	<code>software.amazon.awssdk.services.iotanalytics.IotAnalyticsClient</code>
<code>com.amazonaws.services.iotdata.AWSIoTDataAsyncClient</code>	<code>software.amazon.awssdk.services.iotdata.IotDataAsyncClient</code>
<code>com.amazonaws.services.iotdata.AWSIoTDataClient</code>	<code>software.amazon.awssdk.services.iotdata.IotDataClient</code>
<code>com.amazonaws.services.iotjobsdataplane.AWSIoTJobsDataPlaneAsyncClient</code>	<code>software.amazon.awssdk.services.iotdataplane.IotDataPlaneAsyncClient</code>
<code>com.amazonaws.services.iotjobsdataplane.AWSIoTJobsDataPlaneClient</code>	<code>software.amazon.awssdk.services.iotdataplane.IotDataPlaneClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.kinesis.AmazonKinesisAsyncClient</code>	<code>software.amazon.awssdk.services.kinesis.KinesisAsyncClient</code>
<code>com.amazonaws.services.kinesis.AmazonKinesisClient</code>	<code>software.amazon.awssdk.services.kinesis.KinesisClient</code>
<code>com.amazonaws.services.kinesisanalytics.AmazonKinesisAnalyticsAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisanalytics.KinesisAnalyticsAsyncClient</code>
<code>com.amazonaws.services.kinesisanalytics.AmazonKinesisAnalyticsClient</code>	<code>software.amazon.awssdk.services.kinesisanalytics.KinesisAnalyticsClient</code>
<code>com.amazonaws.services.kinesisfirehose.AmazonKinesisFirehoseAsyncClient</code>	<code>software.amazon.awssdk.services.firehose.FirehoseAsyncClient</code>
<code>com.amazonaws.services.kinesisfirehose.AmazonKinesisFirehoseClient</code>	<code>software.amazon.awssdk.services.firehose.FirehoseClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoArchivedMediaAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisvideoarchivedmedia.KinesisVideoArchivedMediaAsyncClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoArchivedMediaClient</code>	<code>software.amazon.awssdk.services.kinesisvideoarchivedmedia.KinesisVideoArchivedMediaClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisvideo.KinesisVideoAsyncClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoClient</code>	<code>software.amazon.awssdk.services.kinesisvideo.KinesisVideoClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoMediaAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisvideomedia.KinesisVideoMediaAsyncClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoMediaClient</code>	<code>software.amazon.awssdk.services.kinesisvideomedia.KinesisVideoMediaClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoPutMediaClient</code>	Nicht unterstützt
<code>com.amazonaws.services.kms.AWSKMSAsyncClient</code>	<code>software.amazon.awssdk.services.kms.KmsAsyncClient</code>
<code>com.amazonaws.services.kms.AWSKMSClient</code>	<code>software.amazon.awssdk.services.kms.KmsClient</code>
<code>com.amazonaws.services.lambda.AWSLambdaAsyncClient</code>	<code>software.amazon.awssdk.services.lambda.LambdaAsyncClient</code>
<code>com.amazonaws.services.lambda.AWSLambdaClient</code>	<code>software.amazon.awssdk.services.lambda.LambdaClient</code>
<code>com.amazonaws.services.lexmodelbuilding.AmazonLexModelBuildingAsyncClient</code>	<code>software.amazon.awssdk.services.lexmodelbuilding.LexModelBuildingAsyncClient</code>
<code>com.amazonaws.services.lexmodelbuilding.AmazonLexModelBuildingClient</code>	<code>software.amazon.awssdk.services.lexmodelbuilding.LexModelBuildingClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.lexruntime.AmazonLexRuntimeAsyncClient</code>	<code>software.amazon.awssdk.services.lexruntime.LexRuntimeAsyncClient</code>
<code>com.amazonaws.services.lexruntime.AmazonLexRuntimeClient</code>	<code>software.amazon.awssdk.services.lexruntime.LexRuntimeClient</code>
<code>com.amazonaws.services.lightsail.AmazonLightsailAsyncClient</code>	<code>software.amazon.awssdk.services.lightsail.LightsailAsyncClient</code>
<code>com.amazonaws.services.lightsail.AmazonLightsailClient</code>	<code>software.amazon.awssdk.services.lightsail.LightsailClient</code>
<code>com.amazonaws.services.logs.AWSLogsAsyncClient</code>	<code>software.amazon.awssdk.services.logs.LogsAsyncClient</code>
<code>com.amazonaws.services.logs.AWSLogsClient</code>	<code>software.amazon.awssdk.services.logs.LogsClient</code>
<code>com.amazonaws.services.machinelearning.AmazonMachineLearningAsyncClient</code>	<code>software.amazon.awssdk.services.machinelearning.MachineLearningAsyncClient</code>
<code>com.amazonaws.services.machinelearning.AmazonMachineLearningClient</code>	<code>software.amazon.awssdk.services.machinelearning.MachineLearningClient</code>
<code>com.amazonaws.services.macie.AmazonMacieAsyncClient</code>	<code>software.amazon.awssdk.services.macie.MacieAsyncClient</code>
<code>com.amazonaws.services.macie.AmazonMacieClient</code>	<code>software.amazon.awssdk.services.macie.MacieClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsAsyncClient</code>	<code>software.amazon.awssdk.services.marketplacecommerceanalytics.MarketplaceCommerceAnalyticsAsyncClient</code>
<code>com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsClient</code>	<code>software.amazon.awssdk.services.marketplacecommerceanalytics.MarketplaceCommerceAnalyticsClient</code>
<code>com.amazonaws.services.marketplaceentitlement.AWSMarketplaceEntitlementAsyncClient</code>	<code>software.amazon.awssdk.services.marketplaceentitlement.MarketplaceEntitlementAsyncClient</code>
<code>com.amazonaws.services.marketplaceentitlement.AWSMarketplaceEntitlementClient</code>	<code>software.amazon.awssdk.services.marketplaceentitlement.MarketplaceEntitlementClient</code>
<code>com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringAsyncClient</code>	<code>software.amazon.awssdk.services.marketplacemetering.MarketplaceMeteringAsyncClient</code>
<code>com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringClient</code>	<code>software.amazon.awssdk.services.marketplacemetering.MarketplaceMeteringClient</code>
<code>com.amazonaws.services.mediaconvert.AWSMediaConvertAsyncClient</code>	<code>software.amazon.awssdk.services.mediaconvert.MediaConvertAsyncClient</code>
<code>com.amazonaws.services.mediaconvert.AWSMediaConvertClient</code>	<code>software.amazon.awssdk.services.mediaconvert.MediaConvertClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.medialive.AWSMediaLiveAsyncClient</code>	<code>software.amazon.awssdk.services.medialive.MediaLiveAsyncClient</code>
<code>com.amazonaws.services.medialive.AWSMediaLiveClient</code>	<code>software.amazon.awssdk.services.medialive.MediaLiveClient</code>
<code>com.amazonaws.services.mediapackage.AWSMediaPackageAsyncClient</code>	<code>software.amazon.awssdk.services.mediapackage.MediaPackageAsyncClient</code>
<code>com.amazonaws.services.mediapackage.AWSMediaPackageClient</code>	<code>software.amazon.awssdk.services.mediapackage.MediaPackageClient</code>
<code>com.amazonaws.services.mediastore.AWSMediaStoreAsyncClient</code>	<code>software.amazon.awssdk.services.mediastore.MediaStoreAsyncClient</code>
<code>com.amazonaws.services.mediastore.AWSMediaStoreClient</code>	<code>software.amazon.awssdk.services.mediastore.MediaStoreClient</code>
<code>com.amazonaws.services.mediastoredata.AWSMediaStoreDataAsyncClient</code>	<code>software.amazon.awssdk.services.mediastoredata.MediaStoreDataAsyncClient</code>
<code>com.amazonaws.services.mediastoredata.AWSMediaStoreDataClient</code>	<code>software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient</code>
<code>com.amazonaws.services.mediatailor.AWSMediaTailorAsyncClient</code>	<code>software.amazon.awssdk.services.mediatailor.MediaTailorAsyncClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.mediataylor.AWSMediaTailorClient</code>	<code>software.amazon.awssdk.services.mediataylor.MediaTailorClient</code>
<code>com.amazonaws.services.migrationhub.AWSMigrationHubAsyncClient</code>	<code>software.amazon.awssdk.services.migrationhub.MigrationHubAsyncClient</code>
<code>com.amazonaws.services.migrationhub.AWSMigrationHubClient</code>	<code>software.amazon.awssdk.services.migrationhub.MigrationHubClient</code>
<code>com.amazonaws.services.mobile.AWSMobileAsyncClient</code>	<code>software.amazon.awssdk.services.mobile.MobileAsyncClient</code>
<code>com.amazonaws.services.mobile.AWSMobileClient</code>	<code>software.amazon.awssdk.services.mobile.MobileClient</code>
<code>com.amazonaws.services.mq.AmazonMQAsyncClient</code>	<code>software.amazon.awssdk.services.mq.MqAsyncClient</code>
<code>com.amazonaws.services.mq.AmazonMQClient</code>	<code>software.amazon.awssdk.services.mq.MqClient</code>
<code>com.amazonaws.services.mturk.AmazonMTurkAsyncClient</code>	<code>software.amazon.awssdk.services.mturk.MTurkAsyncClient</code>
<code>com.amazonaws.services.mturk.AmazonMTurkClient</code>	<code>software.amazon.awssdk.services.mturk.MTurkClient</code>
<code>com.amazonaws.services.neptune.AmazonNeptuneAsyncClient</code>	<code>software.amazon.awssdk.services.neptune.NeptuneAsyncClient</code>
<code>com.amazonaws.services.neptune.AmazonNeptuneClient</code>	<code>software.amazon.awssdk.services.neptune.NeptuneClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.opsworks.AWSOpsWorksAsyncClient</code>	<code>software.amazon.awssdk.services.opsworks.OpsWorksAsyncClient</code>
<code>com.amazonaws.services.opsworks.AWSOpsWorksClient</code>	<code>software.amazon.awssdk.services.opsworks.OpsWorksClient</code>
<code>com.amazonaws.services.opsworkscm.AWSOpsWorksCMAsyncClient</code>	<code>software.amazon.awssdk.services.opsworkscm.OpsWorksCmAsyncClient</code>
<code>com.amazonaws.services.opsworkscm.AWSOpsWorksCMClient</code>	<code>software.amazon.awssdk.services.opsworkscm.OpsWorksCmClient</code>
<code>com.amazonaws.services.organizations.AWSOrganizationsAsyncClient</code>	<code>software.amazon.awssdk.services.organizations.OrganizationsAsyncClient</code>
<code>com.amazonaws.services.organizations.AWSOrganizationsClient</code>	<code>software.amazon.awssdk.services.organizations.OrganizationsClient</code>
<code>com.amazonaws.services.pi.AWSPIAsyncClient</code>	<code>software.amazon.awssdk.services.pi.PiAsyncClient</code>
<code>com.amazonaws.services.pi.AWSPIClient</code>	<code>software.amazon.awssdk.services.pi.PiClient</code>
<code>com.amazonaws.services.pinpoint.AmazonPinpointAsyncClient</code>	<code>software.amazon.awssdk.services.pinpoint.PinpointAsyncClient</code>
<code>com.amazonaws.services.pinpoint.AmazonPinpointClient</code>	<code>software.amazon.awssdk.services.pinpoint.PinpointClient</code>



1.x-Client	2.x-Client
<code>com.amazonaws.services.polly.AmazonPollyAsyncClient</code>	<code>software.amazon.awssdk.services.polly.PollyAsyncClient</code>
<code>com.amazonaws.services.polly.AmazonPollyClient</code>	<code>software.amazon.awssdk.services.polly.PollyClient</code>
<code>com.amazonaws.services.pricing.AWSPricingAsyncClient</code>	<code>software.amazon.awssdk.services.pricing.PricingAsyncClient</code>
<code>com.amazonaws.services.pricing.AWSPricingClient</code>	<code>software.amazon.awssdk.services.pricing.PricingClient</code>
<code>com.amazonaws.services.rds.AmazonRDSAsyncClient</code>	<code>software.amazon.awssdk.services.rds.RdsAsyncClient</code>
<code>com.amazonaws.services.rds.AmazonRDSClient</code>	<code>software.amazon.awssdk.services.rds.RdsClient</code>
<code>com.amazonaws.services.redshift.AmazonRedshiftAsyncClient</code>	<code>software.amazon.awssdk.services.redshift.RedshiftAsyncClient</code>
<code>com.amazonaws.services.redshift.AmazonRedshiftClient</code>	<code>software.amazon.awssdk.services.redshift.RedshiftClient</code>
<code>com.amazonaws.services.rekognition.AmazonRekognitionAsyncClient</code>	<code>software.amazon.awssdk.services.rekognition.RekognitionAsyncClient</code>
<code>com.amazonaws.services.rekognition.AmazonRekognitionClient</code>	<code>software.amazon.awssdk.services.rekognition.RekognitionClient</code>
<code>com.amazonaws.services.resourcegroups.AWSResourceGroupsAsyncClient</code>	<code>software.amazon.awssdk.services.resourcegroups.ResourceGroupsAsyncClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.resourcegroups.AWSResourceGroupsClient</code>	<code>software.amazon.awssdk.services.resourcegroups.ResourceGroupsClient</code>
<code>com.amazonaws.services.resourcegroupstaggingapi.AWSResourceGroupsTaggingAPIAsyncClient</code>	<code>software.amazon.awssdk.services.resourcegroupstaggingapi.ResourceGroupsTaggingApiAsyncClient</code>
<code>com.amazonaws.services.resourcegroupstaggingapi.AWSResourceGroupsTaggingAPIClient</code>	<code>software.amazon.awssdk.services.resourcegroupstaggingapi.ResourceGroupsTaggingApiClient</code>
<code>com.amazonaws.services.route53.AmazonRoute53AsyncClient</code>	<code>software.amazon.awssdk.services.route53.Route53AsyncClient</code>
<code>com.amazonaws.services.route53.AmazonRoute53Client</code>	<code>software.amazon.awssdk.services.route53.Route53Client</code>
<code>com.amazonaws.services.route53domains.AmazonRoute53DomainsAsyncClient</code>	<code>software.amazon.awssdk.services.route53domains.Route53DomainsAsyncClient</code>
<code>com.amazonaws.services.route53domains.AmazonRoute53DomainsClient</code>	<code>software.amazon.awssdk.services.route53domains.Route53DomainsClient</code>
<code>com.amazonaws.services.s3.AmazonS3Client</code>	<code>software.amazon.awssdk.services.s3.S3Client</code>
<code>com.amazonaws.services.sagemaker.AmazonSageMakerAsyncClient</code>	<code>software.amazon.awssdk.services.sagemaker.SageMakerAsyncClient</code>
<code>com.amazonaws.services.sagemaker.AmazonSageMakerClient</code>	<code>software.amazon.awssdk.services.sagemaker.SageMakerClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.sagemakerruntime.AmazonSageMakerRuntimeAsyncClient</code>	<code>software.amazon.awssdk.services.sagemakerruntime.SageMakerRuntimeAsyncClient</code>
<code>com.amazonaws.services.sagemakerruntime.AmazonSageMakerRuntimeClient</code>	<code>software.amazon.awssdk.services.sagemakerruntime.SageMakerRuntimeClient</code>
<code>com.amazonaws.services.secretsmanager.AWSecretsManagerAsyncClient</code>	<code>software.amazon.awssdk.services.secretsmanager.SecretsManagerAsyncClient</code>
<code>com.amazonaws.services.secretsmanager.AWSecretsManagerClient</code>	<code>software.amazon.awssdk.services.secretsmanager.SecretsManagerClient</code>
<code>com.amazonaws.services.securitytoken.AWSSecurityTokenServiceAsyncClient</code>	<code>software.amazon.awssdk.services.sts.StsAsyncClient</code>
<code>com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClient</code>	<code>software.amazon.awssdk.services.sts.StsClient</code>
<code>com.amazonaws.services.serverlessapplicationrepository.AWSServerlessApplicationRepositoryAsyncClient</code>	<code>software.amazon.awssdk.services.serverlessapplicationrepository.ServerlessApplicationRepositoryAsyncClient</code>
<code>com.amazonaws.services.serverlessapplicationrepository.AWSServerlessApplicationRepositoryClient</code>	<code>software.amazon.awssdk.services.serverlessapplicationrepository.ServerlessApplicationRepositoryClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.servermigration.AWSServerMigrationAsyncClient</code>	<code>software.amazon.awssdk.services.sms.SmsAsyncClient</code>
<code>com.amazonaws.services.servermigration.AWSServerMigrationClient</code>	<code>software.amazon.awssdk.services.sms.SmsClient</code>
<code>com.amazonaws.services.servicecatalog.AWSServiceCatalogAsyncClient</code>	<code>software.amazon.awssdk.services.servicecatalog.ServiceCatalogAsyncClient</code>
<code>com.amazonaws.services.servicecatalog.AWSServiceCatalogClient</code>	<code>software.amazon.awssdk.services.servicecatalog.ServiceCatalogClient</code>
<code>com.amazonaws.services.servicediscovery.AWSServiceDiscoveryAsyncClient</code>	<code>software.amazon.awssdk.services.servicediscovery.ServiceDiscoveryAsyncClient</code>
<code>com.amazonaws.services.servicediscovery.AWSServiceDiscoveryClient</code>	<code>software.amazon.awssdk.services.servicediscovery.ServiceDiscoveryClient</code>
<code>com.amazonaws.services.shield.AWSShieldAsyncClient</code>	<code>software.amazon.awssdk.services.shield.ShieldAsyncClient</code>
<code>com.amazonaws.services.shield.AWSShieldClient</code>	<code>software.amazon.awssdk.services.shield.ShieldClient</code>
<code>com.amazonaws.services.simplesdb.AmazonSimpleDBAsyncClient</code>	<code>software.amazon.awssdk.services.simplesdb.SimpleDbAsyncClient</code>
<code>com.amazonaws.services.simplesdb.AmazonSimpleDBClient</code>	<code>software.amazon.awssdk.services.simplesdb.SimpleDbClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceAsyncClient</code>	<code>software.amazon.awssdk.services.ses.SesAsyncClient</code>
<code>com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceClient</code>	<code>software.amazon.awssdk.services.ses.SesClient</code>
<code>com.amazonaws.services.simplesystemsmanagement.AWSSimpleSystemsManagementAsyncClient</code>	<code>software.amazon.awssdk.services.ssm.SsmAsyncClient</code>
<code>com.amazonaws.services.simplesystemsmanagement.AWSSimpleSystemsManagementClient</code>	<code>software.amazon.awssdk.services.ssm.SsmClient</code>
<code>com.amazonaws.services.simplesworkflow.AmazonSimpleWorkflowAsyncClient</code>	<code>software.amazon.awssdk.services.swf.SwfAsyncClient</code>
<code>com.amazonaws.services.simplesworkflow.AmazonSimpleWorkflowClient</code>	<code>software.amazon.awssdk.services.swf.SwfClient</code>
<code>com.amazonaws.services.snowball.AmazonSnowballAsyncClient</code>	<code>software.amazon.awssdk.services.snowball.SnowballAsyncClient</code>
<code>com.amazonaws.services.snowball.AmazonSnowballClient</code>	<code>software.amazon.awssdk.services.snowball.SnowballClient</code>
<code>com.amazonaws.services.sns.AmazonSNSAsyncClient</code>	<code>software.amazon.awssdk.services.sns.SnsAsyncClient</code>
<code>com.amazonaws.services.sns.AmazonSNSClient</code>	<code>software.amazon.awssdk.services.sns.SnsClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.sqs. AmazonSQSAsyncClient</code>	<code>software.amazon.awssdk.serv ices.sqs.SqsAsyncClient</code>
<code>com.amazonaws.services.sqs. AmazonSQSClient</code>	<code>software.amazon.awssdk.serv ices.sqs.SqsClient</code>
<code>com.amazonaws.services.step functions.AWSStepFunctionsA syncClient</code>	<code>software.amazon.awssdk.serv ices.sfn.SfnAsyncClient</code>
<code>com.amazonaws.services.step functions.AWSStepFunctionsC lient</code>	<code>software.amazon.awssdk.serv ices.sfn.SfnClient</code>
<code>com.amazonaws.services.stor agegateway.AWSStorageGatewa yAsyncClient</code>	<code>software.amazon.awssdk.serv ices.storagegateway.Storage GatewayAsyncClient</code>
<code>com.amazonaws.services.stor agegateway.AWSStorageGatewa yClient</code>	<code>software.amazon.awssdk.serv ices.storagegateway.Storage GatewayClient</code>
<code>com.amazonaws.services.supp ort.AWSSupportAsyncClient</code>	<code>software.amazon.awssdk.serv ices.support.SupportAsyncClient</code>
<code>com.amazonaws.services.supp ort.AWSSupportClient</code>	<code>software.amazon.awssdk.serv ices.support.SupportClient</code>
<code>com.amazonaws.services.tran scribe.AmazonTranscribeAsyn cClient</code>	<code>software.amazon.awssdk.serv ices.transcribe.TranscribeA syncClient</code>
<code>com.amazonaws.services.tran scribe.AmazonTranscribeClient</code>	<code>software.amazon.awssdk.serv ices.transcribe.TranscribeC lient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.translate.AmazonTranslateAsyncClient</code>	<code>software.amazon.awssdk.services.translate.TranslateAsyncClient</code>
<code>com.amazonaws.services.translate.AmazonTranslateClient</code>	<code>software.amazon.awssdk.services.translate.TranslateClient</code>
<code>com.amazonaws.services.waf.AWSWAFAsyncClient</code>	<code>software.amazon.awssdk.services.waf.WafAsyncClient</code>
<code>com.amazonaws.services.waf.AWSWAFClient</code>	<code>software.amazon.awssdk.services.waf.WafClient</code>
<code>com.amazonaws.services.waf.AWSWAFRegionalAsyncClient</code>	<code>software.amazon.awssdk.services.waf.regional.WafRegionalAsyncClient</code>
<code>com.amazonaws.services.waf.AWSWAFRegionalClient</code>	<code>software.amazon.awssdk.services.waf.regional.WafRegionalClient</code>
<code>com.amazonaws.services.workdocs.AmazonWorkDocsAsyncClient</code>	<code>software.amazon.awssdk.services.workdocs.WorkDocsAsyncClient</code>
<code>com.amazonaws.services.workdocs.AmazonWorkDocsClient</code>	<code>software.amazon.awssdk.services.workdocs.WorkDocsClient</code>
<code>com.amazonaws.services.workmail.AmazonWorkMailAsyncClient</code>	<code>software.amazon.awssdk.services.workmail.WorkMailAsyncClient</code>
<code>com.amazonaws.services.workmail.AmazonWorkMailClient</code>	<code>software.amazon.awssdk.services.workmail.WorkMailClient</code>

1.x-Client	2.x-Client
<code>com.amazonaws.services.workspaces.AmazonWorkspacesAsyncClient</code>	<code>software.amazon.awssdk.services.workspaces.WorkSpacesAsyncClient</code>
<code>com.amazonaws.services.workspaces.AmazonWorkspacesClient</code>	<code>software.amazon.awssdk.services.workspaces.WorkSpacesClient</code>
<code>com.amazonaws.services.xray.AWSXRayAsyncClient</code>	<code>software.amazon.awssdk.services.xray.XRayAsyncClient</code>
<code>com.amazonaws.services.xray.AWSXRayClient</code>	<code>software.amazon.awssdk.services.xray.XRayClient</code>

## Standardeinstellungen für die Client-Erstellung

In Version 2.x wurden die folgenden Änderungen an der Standardlogik für die Client-Erstellung vorgenommen.

- Die standardmäßige Anbieterkette für Anmeldeinformationen für S3 enthält keine anonymen Anmeldeinformationen mehr. Sie müssen den anonymen Zugriff auf S3 manuell angeben, indem Sie die `useAnonCredentialsProvider` verwenden.
- Die folgenden Umgebungsvariablen im Zusammenhang mit der Standard-Client-Erstellung unterscheiden sich.

1.x	2.x
<code>AWS_CBOR_DISABLED</code>	<code>CBOR_ENABLED</code>
<code>AWS_ION_BINARY_DISABLE</code>	<code>BINARY_ION_ENABLED</code>

- Die folgenden Systemeigenschaften im Zusammenhang mit der Standard-Client-Erstellung unterscheiden sich.



1.x	2.x
<code>com.amazonaws.sdk.disableEc2Metadata</code>	<code>aws.disableEc2Metadata</code>
<code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	<code>aws.ec2MetadataServiceEndpoint</code>
<code>com.amazonaws.sdk.disableCbor</code>	<code>aws.cborEnabled</code>
<code>com.amazonaws.sdk.disableIoBinary</code>	<code>aws.binaryIonEnabled</code>

- Version 2.x unterstützt die folgenden Systemeigenschaften nicht.

- 

1.x
<code>com.amazonaws.sdk.disableCertChecking</code>
<code>com.amazonaws.sdk.enableDefaultMetrics</code>
<code>com.amazonaws.sdk.enableThrottledRetry</code>
<code>com.amazonaws.regions.RegionUtils.fileOverride</code>
<code>com.amazonaws.regions.RegionUtils.disableRemote</code>
<code>com.amazonaws.services.s3.disableImplicitGlobalClients</code>
<code>com.amazonaws.sdk.enableInRegionOptimizedMode</code>

- Das Laden der Regionskonfiguration aus einer benutzerdefinierten `endpoints.json` Datei wird nicht mehr unterstützt.

## Client-Konfiguration

In 1.x wurde die SDK-Client-Konfiguration geändert, indem eine `ClientConfiguration` Instance auf dem Client oder Client Builder festgelegt wurde. In Version 2.x wird die Client-Konfiguration in getrennte Konfigurationsklassen aufgeteilt. Mit den separaten Konfigurationsklassen können Sie

verschiedene HTTP-Clients für asynchrone und synchrone Clients konfigurieren, aber immer noch dieselbe `ClientOverrideConfiguration` Klasse verwenden.

### Example der Client-Konfiguration in Version 1.x

```
AmazonDynamoDBClientBuilder.standard()  
.withClientConfiguration(clientConfiguration)  
.build()
```

### Example Synchrone Client-Konfiguration in Version 2.x

```
ProxyConfiguration.Builder proxyConfig = ProxyConfiguration.builder();  
  
ApacheHttpClient.Builder httpClientBuilder =  
    ApacheHttpClient.builder()  
        .proxyConfiguration(proxyConfig.build());  
  
ClientOverrideConfiguration.Builder overrideConfig =  
    ClientOverrideConfiguration.builder();  
  
DynamoDbClient client =  
    DynamoDbClient.builder()  
        .httpClientBuilder(httpClientBuilder)  
        .overrideConfiguration(overrideConfig.build())  
        .build();
```

### Example Asynchrone Client-Konfiguration in Version 2.x

```
NettyNioAsyncHttpClient.Builder httpClientBuilder =  
    NettyNioAsyncHttpClient.builder();  
  
ClientOverrideConfiguration.Builder overrideConfig =  
    ClientOverrideConfiguration.builder();  
  
ClientAsyncConfiguration.Builder asyncConfig =  
    ClientAsyncConfiguration.builder();  
  
DynamoDbAsyncClient client =  
    DynamoDbAsyncClient.builder()  
        .httpClientBuilder(httpClientBuilder)  
        .overrideConfiguration(overrideConfig.build())  
        .asyncConfiguration(asyncConfig.build())
```

```
.build();
```

## HTTP-Clients

### Wichtige Änderungen

- In Version 2.x können Sie ändern, welcher HTTP-Client zur Laufzeit verwendet werden soll, indem Sie eine Implementierung mit `angebenclientBuilder.httpClientBuilder`.
- Wenn Sie einen HTTP-Client mithilfe von `clientBuilder.httpClient` an einen Service-Client-BUILDER übergeben, wird der HTTP-Client nicht standardmäßig geschlossen, wenn der Service-Client geschlossen wird. Auf diese Weise können Sie HTTP-Clients zwischen Service-Clients teilen.
- Asynchrone HTTP-Clients verwenden jetzt nicht blockierendes IO.
- Einige Operationen verwenden jetzt HTTP/2 für eine verbesserte Leistung.

### Änderungen an den Einstellungen

Einstellung	1.x	2.x Sync, Apache	2.x Async, Netty
	<pre>ClientCon figuration clientConfig =     new ClientCon figuration()</pre>	<pre>ApacheHtt pClient.B uilder httpClien tBuilder =     ApacheHtt pClient.b uilder()</pre>	<pre>NettyNioA syncHttpC lient.Builder httpClient tBuilder =     NettyNioA syncHttpC lient.builder()</pre>
Maximale Verbindun gen	<pre>clientCon fig.setMa xConnecti ons(...) clientCon fig.withM axConnect ions(...)</pre>	<pre>httpClien tBuilder. maxConnec tions(...)</pre>	<pre>httpClien tBuilder. maxConcur rency(...)</pre>

Einstellung	1.x	2.x Sync, Apache	2.x Async, Netty
Verbindungstimeout	<pre>clientConfig.setConnectionTimeout(...) clientConfig.withConnectionTimeout(...)</pre>	<pre>httpClientBuilder.connectionTimeout(...)</pre>	<pre>httpClientBuilder.connectionTimeout(...)</pre>
Socket-Timeout	<pre>clientConfig.setSocketTimeout(...) clientConfig.withSocketTimeout(...)</pre>	<pre>httpClientBuilder.socketTimeout(...)</pre>	<pre>httpClientBuilder.writeTimeout(...) httpClientBuilder.readTimeout(...)</pre>
Verbindungs-TTL	<pre>clientConfig.setConnectionTTL(...) clientConfig.withConnectionTTL(...)</pre>	<pre>httpClientBuilder.connectionTimeToLive(...)</pre>	<pre>httpClientBuilder.connectionTimeToLive(...)</pre>
Maximale Leerlaufzeit der Verbindung	<pre>clientConfig.setConnectionMaxIdleMillis(...) clientConfig.withConnectionMaxIdleMillis(...)</pre>	<pre>httpClientBuilder.connectionMaxIdleTime(...)</pre>	<pre>httpClientBuilder.connectionMaxIdleTime(...)</pre>

Einstellung	1.x	2.x Sync, Apache	2.x Async, Netty
Nach Inaktivität validieren	<pre>clientConfig.setValidateAfterInactivityMillis(...) clientConfig.withValidateAfterInactivityMillis(...)</pre>	Nicht unterstützt ( <a href="#">Funktion anfordern</a> )	Nicht unterstützt ( <a href="#">Anforderungsfunktion</a> )
Lokale Adresse	<pre>clientConfig.setLocalAddress(...) clientConfig.withLocalAddress(...)</pre>	<pre>httpClientBuilder.localAddress(...)</pre>	<a href="#">Nicht unterstützt</a>
Fortsetzung erwarten aktiviert	<pre>clientConfig.setUseExpectContinue(...) clientConfig.withUseExpectContinue(...)</pre>	<pre>httpClientBuilder.expectContinueEnabled(...)</pre>	Nicht unterstützt ( <a href="#">Anforderungsfunktion</a> )
Verbindungsaufwand	<pre>clientConfig.setUseReaper(...) clientConfig.withReaper(...)</pre>	<pre>httpClientBuilder.useIdleConnectionReaper(...)</pre>	<pre>httpClientBuilder.useIdleConnectionReaper(...)</pre>

Einstellung	1.x	2.x Sync, Apache	2.x Async, Netty
	<pre>AmazonDynamoDBClientBuilder .standard() .withClientConfiguration( clientConfiguration) .build()</pre>	<pre>DynamoDBClient.builder() .httpClientBuilder( httpClientBuilder) .build()</pre>	<pre>DynamoDBAsyncClient.builder() .httpClientBuilder( httpClientBuilder) .build()</pre>

## HTTP-Client-Proxys

Einstellungen	1.x	2.x Sync, Apache	2.x Async, Netty
	<pre>ClientConfiguration clientConfig = new ClientConfiguration()</pre>	<pre>ProxyConfiguration .Builder proxyConfig = ProxyConfiguration .builder()</pre>	<pre>ProxyConfiguration .Builder proxyConfig = ProxyConfiguration .builder()</pre>
Proxy-Host	<pre>clientConfig.setProxyHost(...) clientConfig.withProxyHost(...)</pre>	<pre>proxyConfig.endpoint(...)</pre>	<pre>proxyConfig.host(...)</pre>
Proxy-Port	<pre>clientConfig.setProxyPort(...) clientConfig.withProxyPort(...)</pre>	<pre>proxyConfig.endpoint(...)</pre>	<pre>proxyConfig.port(...)</pre>

Einstellungen	1.x	2.x Sync, Apache	2.x Async, Netty
		<u>Proxy-Port</u> ist eingebettet in endpoint	
Proxy-Benutzername	<pre>clientConfig.setProxyUsername(...) clientConfig.withProxyUsername(...)</pre>	<pre>proxyConfig.username(...)</pre>	<pre>proxyConfig.username(...)</pre>
Proxy-Passwort	<pre>clientConfig.setProxyPassword(...) clientConfig.withProxyPassword(...)</pre>	<pre>proxyConfig.password(...)</pre>	<pre>proxyConfig.password(...)</pre>
Proxy-Domäne	<pre>clientConfig.setProxyDomain(...) clientConfig.withProxyDomain(...)</pre>	<pre>proxyConfig.ntlmDomain(...)</pre>	Nicht unterstützt ( <a href="#">Anforderungsfunktion</a> )
Proxy-Workstation	<pre>clientConfig.setProxyWorkspace(...) clientConfig.withProxyWorkstation(...)</pre>	<pre>proxyConfig.ntlmWorkstation(...)</pre>	Nicht unterstützt ( <a href="#">Anforderungsfunktion</a> )

Einstellungen	1.x	2.x Sync, Apache	2.x Async, Netty
Proxy-Authentifizierungsmethoden	<pre>clientConfig.setProxyAuthenticationMethods(...) clientConfig.withProxyAuthenticationMethods(...)</pre>	<u>Nicht unterstützt</u>	Nicht unterstützt ( <a href="#">Anforderungsfunktion</a> )
Präemptive grundlegende Proxy-Authentifizierung	<pre>clientConfig.setPreemptiveBasicProxyAuth(...) clientConfig.withPreemptiveBasicProxyAuth(...)</pre>	<pre>proxyConfig.preemptiveBasicAuthenticationEnabled(...)</pre>	Nicht unterstützt ( <a href="#">Anforderungsfunktion</a> )
Nicht-Proxy-Hosts	<pre>clientConfig.setNonProxyHosts(...) clientConfig.withNonProxyHosts(...)</pre>	<pre>proxyConfig.nonProxyHosts(...)</pre>	<pre>proxyConfig.nonProxyHosts(...)</pre>



Einstellungen	1.x	2.x Sync, Apache	2.x Async, Netty
Socket-Proxy deaktivieren	<pre>clientConfig.setDisableSocketProxy(...) clientConfig.withDisableSocketProxy(...)</pre>	Nicht unterstützt ( <a href="#">Anforderungsfunktion</a> )	Nicht unterstützt ( <a href="#">Anforderungsfunktion</a> )
	<pre>AmazonDynamoDBClientBuilder     .standard()     .withClientConfiguration(clientConfiguration)     .build()</pre>	<pre>httpClientBuilder.proxyConfiguration(proxyConfiguration).build()</pre>	<pre>httpClientBuilder.proxyConfiguration(proxyConfiguration).build()</pre>

### Client-Überschreibungen

Einstellung	1.x	2.x
	<pre>ClientConfiguration clientConfig = new ClientConfiguration()</pre>	<pre>ClientOverrideConfiguration.Builder overrideConfig = ClientOverrideConfiguration.builder()</pre>
Benutzeragentenpräfix	<pre>clientConfig.setUserAgentPrefix(...) clientConfig.withUserAgentPrefix(...)</pre>	<pre>overrideConfig.advancedOption(SdkAdvancedClientOption.USER_AGENT_PREFIX, ...)</pre>

Einstellung	1.x	2.x
Suffix des Benutzeragenten	<pre>clientConfig.setUserAgentSuffix(...) clientConfig.withUserAgentSuffix(...)</pre>	<pre>overrideConfig.advancedOption(     SdkAdvancedClientOption.USER_AGENT_SUFFIX, ...)</pre>
Signer	<pre>clientConfig.setSignerOverride(...) clientConfig.withSignerOverride(...)</pre>	<pre>overrideConfig.advancedOption(     SdkAdvancedClientOption.SIGNER, ...)</pre>
Zusätzliche Header	<pre>clientConfig.addHeader(...) clientConfig.withHeader(...)</pre>	<pre>overrideConfig.putHeader(...)</pre>
Anforderungs-Timeout	<pre>clientConfig.setRequestTimeout(...) clientConfig.withRequestTimeout(...)</pre>	<pre>overrideConfig.apiCallAttemptTimeout(...)</pre>
Client-Ausführungs-Timeout	<pre>clientConfig.setClientExecutionTimeout(...) clientConfig.withClientExecutionTimeout(...)</pre>	<pre>overrideConfig.apiCallTimeout(...)</pre>
Verwenden von Gzip	<pre>clientConfig.setUseGzip(...) clientConfig.withGzip(...)</pre>	Nicht unterstützt ( <a href="#">Anforderungsfunktion</a> )

Einstellung	1.x	2.x
Hinweis zur Socket-Puffergröße	<pre>clientConfig.setSocketBufferSizeHints(...) clientConfig.withSocketBufferSizeHints(...)</pre>	Nicht unterstützt ( <a href="#">Anforderungsfunktion</a> )
Cache-Antwortmetadaten	<pre>clientConfig.setCacheResponseMetadata(...) clientConfig.withCacheResponseMetadata(...)</pre>	Nicht unterstützt ( <a href="#">Anforderungsfunktion</a> )
Größe des Antwortmetadaten-Cache	<pre>clientConfig.setResponseMetadataCacheSize(...) clientConfig.withResponseMetadataCacheSize(...)</pre>	Nicht unterstützt ( <a href="#">Anforderungsfunktion</a> )
DNS-Auflöser	<pre>clientConfig.setDnsResolver(...) clientConfig.withDnsResolver(...)</pre>	Nicht unterstützt ( <a href="#">Anforderungsfunktion</a> )
TCP-Keepalive	<pre>clientConfig.setUseTcpKeepAlive(...) clientConfig.withTcpKeepAlive(...)</pre>	<p>Diese Option befindet sich jetzt in der HTTP-Client-Konfiguration</p> <ul style="list-style-type: none"> <li>- <code>ApacheHttpClient.builder().tcpKeepAlive(true)</code></li> <li>- <code>NettyNioAsyncHttpClient.builder().tcpKeepAlive(true)</code></li> </ul>

Einstellung	1.x	2.x
Sicheres Zufallsergebnis	<pre>clientConfig.setSecureRandom(...) clientConfig.withSecureRandom(...)</pre>	Nicht unterstützt ( <a href="#">Anforderungsfunktion</a> )
	<pre>AmazonDynamoDBClientBuilder.standard()     .withClientConfiguration(clientConfiguration)     .build()</pre>	<pre>DynamoDbClient.builder()     .httpClientBuilder(httpClientBuilder)     .build()</pre>

### Wiederholungsversuche bei der Client-Überschreibung

Einstellung	1.x	2.x
	<pre>ClientConfiguration clientConfig =     new ClientConfiguration()</pre>	<pre>RetryPolicy.Builder retryPolicy =     RetryPolicy.builder()</pre>
Maximale Fehlerwiederholung	<pre>clientConfig.setMaxErrorRetry(...) clientConfig.withMaxErrorRetry(...)</pre>	<pre>retryPolicy.numRetries(...)</pre>
Drosselte Wiederholungen verwenden	<pre>clientConfig.setUseThrottleRetries(...) clientConfig.withUseThrottleRetries(...)</pre>	<a href="#">Nicht unterstützt</a>

Einstellung	1.x	2.x
Maximale Anzahl aufeinanderfolgender Wiederholungen vor der Drosselung	<pre>clientConfig.setMaxConsecutiveRetriesBeforeThrottling(...) clientConfig.withMaxConsecutiveRetriesBeforeThrottling(...)</pre>	<u>Nicht unterstützt</u>
	<pre>AmazonDynamoDBClientBuilder.standard()     .withClientConfiguration(clientConfiguration)     .build()</pre>	<pre>DynamoDbClient.builder()     .httpClientBuilder(httpClientBuilder)     .build()</pre>

## Asynchrone Clients

Einstellung	1.x	2.x
		<pre>ClientAsyncConfiguration.Builder     asyncConfig =         ClientAsyncConfiguration.builder()</pre>
Executor	<pre>AmazonDynamoDBAsyncClientBuilder.standard()     .withExecutorFactory(...)     .build()</pre>	<pre>asyncConfig.advancedOption(     SdkAdvancedAsyncClientOption.FUTURE_COMPLETION_EXECUTOR, ...)</pre>
		<pre>DynamoDbAsyncClient.builder()</pre>

Einstellung	1.x	2.x
		<pre>.asyncConfiguration(     asyncConfig)     .build()</pre>

## Andere Client-Änderungen

Die folgende `ClientConfiguration` Option von 1.x hat sich in 2.x des SDK geändert und hat kein direktes Äquivalent.

Einstellung	1.x	2.x-Äquivalent
Protokoll	<pre>clientConfig.setProtocol(     Protocol.HTTP) clientConfig.withProtocol(     Protocol.HTTP)</pre>	<p>Die Protokolleinstellung ist standardmäßig HTTPS. Um die Einstellung zu ändern, geben Sie die Protokolleinstellung für einen HTTP-Endpoint im Client Builder an:</p> <pre>clientBuilder.endpointOverride(     URI.create("http://..."))</pre>

## Änderungen des Anmeldeinformationsanbieters

Dieser Abschnitt enthält eine Zuordnung der Namensänderungen von Anmeldeinformationsanbieterklassen und -methoden zwischen den Versionen 1.x und 2.x des AWS SDK for Java.

### Bemerkenswerte Unterschiede

- Der Standard-Anmeldeinformationsanbieter lädt in Version 2.x Systemeigenschaften vor Umgebungsvariablen. Weitere Informationen finden Sie unter [Verwenden von Anmeldeinformationen](#).
- Die Konstruktormethode wurde durch die `create`- oder `builder`-Methoden ersetzt.

## Example

```
DefaultCredentialsProvider.create();
```

- Asynchrone Aktualisierung wird nicht mehr standardmäßig festgelegt. Sie müssen sie mit dem `builder` des Anmeldeinformationsanbieters angeben.

## Example

```
ContainerCredentialsProvider provider = ContainerCredentialsProvider.builder()
    .asyncCredentialUpdateEnabled(true)
    .build();
```

- Sie können mit dem `ProfileCredentialsProvider.builder()` einen Pfad zu einer benutzerdefinierten Profildatei angeben.

## Example

```
ProfileCredentialsProvider profile = ProfileCredentialsProvider.builder()
    .profileFile(ProfileFile.builder().content(Paths.get("myProfileFile.file")).build())
    .build();
```

- Das Profildateiformat wurde geändert, um eine größere Übereinstimmung mit der AWS CLI zu erreichen. Weitere Informationen finden Sie unter [Konfigurieren der AWS CLI](#) im AWS Command Line Interface -Benutzerhandbuch.

Änderungen des Anmeldeinformationsanbieters, die zwischen den Versionen 1.x und 2.x zugeordnet sind

## AWSCredentialsProvider

Kategorie ändern	1.x	2.x
Paket-/Klassenname	<code>com.amazonaws.auth.AWSCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.AwsCredentialsProvider</code>

Kategorie ändern	1.x	2.x
Methodenname	<code>getCredentials</code>	<code>resolveCredentials</code>
Nicht unterstützte Methode	<code>refresh</code>	Nicht unterstützt

## DefaultAWSCredentialsProviderChain

Kategorie ändern	1.x	2.x
Paket-/Klassenname	<code>com.amazonaws.auth.DefaultAWSCredentialsProviderChain</code>	<code>software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider</code>
Erstellung	<code>new DefaultAWSCredentialsProviderChain</code>	<code>DefaultCredentialsProvider.create</code>
Nicht unterstützte Methode	<code>getInstance</code>	Nicht unterstützt
Prioritätsreihenfolge externer Einstellungen	Umgebungsvariablen vor Systemeigenschaften	Systemeigenschaften vor Umgebungsvariablen

## AWSStaticCredentialsProvider

Kategorie ändern	1.x	2.x
Paket-/Klassenname	<code>com.amazonaws.auth.AWSStaticCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.StaticCredentialsProvider</code>
Erstellung	<code>new AWSStaticCredentialsProvider</code>	<code>StaticCredentialsProvider.create</code>



**EnvironmentVariableCredentialsProvider**

Kategorie ändern	1.x	2.x
Paket-/Klassenname	<code>com.amazonaws.auth.EnvironmentVariableCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider</code>
Erstellung	<code>new EnvironmentVariableCredentialsProvider</code>	<code>EnvironmentVariableCredentialsProvider.create</code>
Name der Umgebungsvariablen	<code>AWS_ACCESS_KEY</code>	<code>AWS_ACCESS_KEY_ID</code>
	<code>AWS_SECRET_KEY</code>	<code>AWS_SECRET_ACCESS_KEY</code>

**SystemPropertiesCredentialsProvider**

Kategorie ändern	1.x	2.x
Paket-/Klassenname	<code>com.amazonaws.auth.SystemPropertiesCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.SystemPropertyCredentialsProvider</code>
Erstellung	<code>new SystemPropertiesCredentialsProvider</code>	<code>SystemPropertiesCredentialsProvider.create</code>
Name der Systemeigenschaft	<code>aws.secretKey</code>	<code>aws.secretAccessKey</code>

## ProfileCredentialsProvider

Kategorie ändern	1.x	2.x
Paket-/Klassenname	<code>com.amazonaws.auth .profile.ProfileCr edentialsProvider</code>	<code>software.amazon.aw ssdk.auth.credenti als.ProfileCredent ialsProvider</code>
Erstellung	<code>new ProfileCr edentialsProvider</code>	<code>ProfileCredentials Provider.create</code>
Speicherort des benutzerd efinierten Profils	<ul style="list-style-type: none"> <li>• <code>AWS_CREDENTIAL_PRO FILES_FILE</code> Umgebungsvariable</li> <li>• <code>new ProfileCr edentialsProvider</code></li> </ul>	<ul style="list-style-type: none"> <li>• <code>AWS_SHARED_CREDENT IALS_FILE</code> Umgebungs variable</li> <li>• <code>ProfileCredentials Provider.builder</code></li> </ul>

## ContainerCredentialsProvider

Kategorie ändern	1.x	2.x
Paket-/Klassenname	<code>com.amazonaws.auth .ContainerCredenti alsProvider</code>	<code>software.amazon.aw ssdk.auth.credenti als.ContainerCrede ntialsProvider</code>
Erstellung	<code>new Container CredentialsProvider</code>	<code>ContainerCredentia lsProvider.create</code>
Asynchrone Aktualisierung angeben	Nicht unterstützt	Standardverhalten

## InstanceProfileCredentialsProvider

Kategorie ändern	1.x	2.x
Paket-/Klassenname	<code>com.amazonaws.auth.InstanceProfileCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider</code>
Erstellung	<code>new InstanceProfileCredentialsProvider</code>	<code>InstanceProfileCredentialsProvider.create</code>
Asynchrone Aktualisierung angeben	<code>new InstanceProfileCredentialsProvider(true)</code>	<code>InstanceProfileCredentialsProvider.builder().asyncCredentialUpdateEnabled(true).build()</code>
Name der Systemeigenschaft	<code>com.amazonaws.sdk.disableEc2Metadata</code>	<code>aws.disableEc2Metadata</code>
	<code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	<code>aws.ec2MetadataServiceEndpoint</code>

## STSAssumeRoleSessionCredentialsProvider

Kategorie ändern	1.x	2.x
Paket-/Klassenname	<code>com.amazonaws.auth.STSAssumeRoleSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsAssumeRoleCredentialsProvider</code>

Kategorie ändern	1.x	2.x
Erstellung	<ul style="list-style-type: none"> <li>• <code>new STSAssumeRoleSessionCredentialsProvider</code></li> <li>• <code>new STSAssumeRoleSessionCredentialsProvider.Builder</code></li> </ul>	<code>StsAssumeRoleCredentialsProvider.builder</code>
Asynchrone Aktualisierung	Standardverhalten	Standardverhalten
Konfiguration	<code>new STSAssumeRoleSessionCredentialsProvider.Builder</code>	Konfigurieren einer - <code>StsClient</code> und - <code>AssumeRoleRequest</code> Anforderung

## STSSessionCredentialsProvider

Kategorie ändern	1.x	2.x
Paket-/Klassenname	<code>com.amazonaws.auth.STSSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsGetSessionTokenCredentialsProvider</code>
Erstellung	<code>new STSAssumeRoleSessionCredentialsProvider</code>	<code>StsGetSessionTokenCredentialsProvider.builder</code>
Asynchrone Aktualisierung	Standardverhalten	<code>StsGetSessionTokenCredentialsProvider.builder</code>
Konfiguration	Konstruktor-Parameter	Konfigurieren einer - <code>StsClient</code> und

Kategorie ändern	1.x	2.x
		-GetSessionTokenRequest Anforderung in einem Builder

## WebIdentityFederationSessionCredentialsProvider

Kategorie ändern	1.x	2.x
Paket-/Klassenname	com.amazonaws.auth. WebIdentityFederationSessionCredentialsProvider	software.amazon.awssdk.services.sts.auth.StsAssumeRoleWithWebIdentityCredentialsProvider
Erstellung	new WebIdentityFederationSessionCredentialsProvider	StsAssumeRoleWithWebIdentityCredentialsProvider.builder
Asynchrone Aktualisierung	Standardverhalten	StsAssumeRoleWithWebIdentityCredentialsProvider.builder
Konfiguration	Konstruktor-Parameter	Konfigurieren einer - StsClient und -AssumeRoleWithWebIdentityRequest Anforderung in einem Builder

## Ersetzte Klassen

1.x-Klasse	2.x Ersatzklassen
<code>com.amazonaws.auth.EC2ContainerCredentialsProviderWrapper</code>	<code>software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider</code> und <code>software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider</code>
<code>com.amazonaws.services.s3.S3CredentialsProviderChain</code>	<code>software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider</code> und <code>software.amazon.awssdk.auth.credentials.AnonymousCredentialsProvider</code>

## Klassen entfernt

1.x-Klasse
<code>com.amazonaws.auth.ClasspathPropertiesFileCredentialsProvider</code>
<code>com.amazonaws.auth.PropertiesFileCredentialsProvider</code>

## Änderungen der Region

In diesem Abschnitt werden die im AWS SDK for Java 2.x implementierten Änderungen für die Verwendung der Regions Klassen `Region` und beschrieben.

### Konfiguration der Region

- Einige AWS Services verfügen nicht über regionsspezifische Endpunkte. Bei Verwendung dieser Services müssen Sie die Region als `Region.AWS_GLOBAL` oder `Region.AWS_CN_GLOBAL` festlegen.

## Example

```
Region region = Region.AWS_GLOBAL;
```

- Die Klassen `com.amazonaws.regions.Regions` und `com.amazonaws.regions.Region` wurden nun in eine Klasse `software.amazon.awssdk.regions.Region` kombiniert.

## Methoden- und Klassennamenzuordnungen

In den folgenden Tabellen werden regionsbezogene Klassen zwischen den Versionen 1.x und 2.x des zugeordnet AWS SDK for Java. Sie können eine Instance dieser Klassen mit der `of()`-Methode erstellen.

### Example

```
RegionMetadata regionMetadata = RegionMetadata.of(Region.US_EAST_1);
```

### Änderungen der 1.x-Regionen-Klassenmethode

1.x	2.x
<code>Regions.fromName</code>	<code>Region.of</code>
<code>Regions.getName</code>	<code>Region.id</code>
<code>Regions.getDescription</code>	<code>Region.metadata().description()</code>
<code>Regions.getCurrentRegion</code>	Nicht unterstützt
<code>Regions.DEFAULT_REGION</code>	Nicht unterstützt
<code>Regions.name</code>	<code>Region.id</code>

### Änderungen der Regionsklassenmethode 1.x

1.x	2.x
<code>Region.getName</code>	<code>Region.id</code>

1.x	2.x
<code>Region.hasHttpsEndpoint</code>	Nicht unterstützt
<code>Region.hasHttpEndpoint</code>	Nicht unterstützt
<code>Region.getAvailableEndpoints</code>	Nicht unterstützt
<code>Region.createClient</code>	Nicht unterstützt

### RegionMetadata Änderungen der -Klassenmethode

1.x	2.x
<code>RegionMetadata.getName</code>	<code>RegionMetadata.name</code>
<code>RegionMetadata.getDomain</code>	<code>RegionMetadata.domain</code>
<code>RegionMetadata.getPartition</code>	<code>RegionMetadata.partition</code>

### ServiceMetadata Änderungen der -Klassenmethode

1.x	2.x
<code>Region.getServiceEndpoint</code>	<code>ServiceMetadata.endpointFor(Region)</code>
<code>Region.isServiceSupported</code>	<code>ServiceMetadata.regions().contains(Region)</code>

## Änderungen an Vorgängen, Anforderungen und Antworten

In v2.x des SDK for Java werden Anforderungen an eine Client-Operation übergeben. Beispielsweise `DynamoDbClient`'s `PutItemRequest` wird an den `-DynamoDbClient.putItemVorgang` übergeben. Diese Operationen geben eine Antwort von zurückAWS-Service, z. B. `PutItemResponse`.

Version 2.x des SDK for Java hat die folgenden Änderungen gegenüber 1.x.



- Operationen mit mehreren Antwortseiten verfügen jetzt über eine `Paginator` Methode zum automatischen Iterieren aller Elemente in der Antwort.
- Sie können Anfragen und Antworten nicht mutieren.
- Sie müssen Anforderungen und Antworten mit einer statischen Builder-Methode anstelle eines Konstruktors erstellen. Beispielsweise `new PutItemRequest().withTableName(...)` ist 1.x jetzt `PutItemRequest.builder().tableName(...).build()`.
- Operationen unterstützen eine Kurznotation zum Erstellen von Anforderungen:  
`dynamoDbClient.putItem(request -> request.tableName(...))`.

## Streaming-Operationen

Streaming-Operationen wie Amazon S3 `getObject` und `putObject` Methoden unterstützen jetzt nicht blockierende E/A. Daher nehmen die Anforderungs- und Antwort-POJOs kein mehr `InputStream` als Parameter an. Stattdessen akzeptiert das Anforderungsobjekt für synchrone Anforderungen `RequestBody`, einen Stream von Bytes. Das asynchrone Äquivalent akzeptiert ein `AsyncRequestBody`.

### Example der Amazon S3-**putObject**Operation in 1.x

```
s3client.putObject(BUCKET, KEY, new File(file_path));
```

### Example der Amazon S3-**putObject**Operation in 2.x

```
s3client.putObject(PutObjectRequest.builder()
    .bucket(BUCKET)
    .key(KEY)
    .build(),
    RequestBody.of(Paths.get("myfile.in")));
```

Parallel akzeptiert ein Streaming-Antwortobjekt eine `ResponseTransformer` für synchrone Clients und eine `AsyncResponseTransformer` für asynchrone Clients.

### Example der Amazon S3-**getObject**Operation in 1.x

```
S3Object o = s3.getObject(bucket, key);
S3ObjectInputStream s3is = o.getObjectContent();
FileOutputStream fos = new FileOutputStream(new File(key));
```

## Example der Amazon S3-`getObject`Operation in 2.x

```
s3client.getObject(GetObjectRequest.builder().bucket(bucket).key(key).build(),
    ResponseTransformer.toFile(Paths.get("key")));
```

Im SDK for Java 2.x verfügen Streaming-Antwortoperationen über eine `AsBytes` Methode zum Laden der Antwort in den Speicher und vereinfachen gängige In-Memory-Typkonvertierungen.

## Ausnahmeänderungen

Namen von Ausnahmeklassen, ihre Strukturen und ihre Beziehungen haben sich geändert. `software.amazon.awssdk.core.exception.SdkException` ist die neue `Exception` Basisklasse, die alle anderen Ausnahmen erweitern.

In dieser Tabelle werden die Ausnahmeänderungen der Klassennamen zugewiesen.

1.x	2.x
<code>com.amazonaws.SdkBaseException</code> <code>com.amazonaws.AmazonClientException</code>	<code>software.amazon.awssdk.core.exception.SdkException</code>
<code>com.amazonaws.SdkClientException</code>	<code>software.amazon.awssdk.core.exception.SdkClientException</code>
<code>com.amazonaws.AmazonServiceException</code>	<code>software.amazon.awssdk.awscore.exception.AwsServiceException</code>

In der folgenden Tabelle werden die Methoden für Ausnahmeklassen zwischen Version 1.x und 2.x zugeordnet.

1.x	2.x
<code>AmazonServiceException.getRequestId</code>	<code>SdkServiceException.requestId</code>

1.x	2.x
<code>AmazonServiceException.getServiceName</code>	<code>AwsServiceException.awsErrorDetails().serviceName</code>
<code>AmazonServiceException.getErrorCode</code>	<code>AwsServiceException.awsErrorDetails().errorCode</code>
<code>AmazonServiceException.getErrorMessage</code>	<code>AwsServiceException.awsErrorDetails().errorMessage</code>
<code>AmazonServiceException.getStatusCode</code>	<code>AwsServiceException.awsErrorDetails().sdkHttpResponse().statusCode</code>
<code>AmazonServiceException.getHttpHeaders</code>	<code>AwsServiceException.awsErrorDetails().sdkHttpResponse().headers</code>
<code>AmazonServiceException.getRawResponse</code>	<code>AwsServiceException.awsErrorDetails().rawResponse</code>

## Serialisierungsänderungen

Das SDK für Java v1.x und v2.x unterscheiden sich in der Art und Weise, wie sie Objekte auflisten serialisieren, um Parameter anzufordern.

Das SDK for Java 1.x serialisiert keine leere Liste, während das SDK for Java 2.x eine leere Liste als leeren Parameter serialisiert.

Betrachten Sie beispielsweise einen Service mit einem `SampleOperation`, der ein `verwendetSampleRequest` verwendet. Der `SampleRequest` akzeptiert zwei Parameter – einen Zeichenfolgentyp `str1` und einen Listentyp `listParam`– wie in den folgenden Beispielen gezeigt.

Example von **SampleOperation** in 1.x

```
SampleRequest v1Request = new SampleRequest()
    .withStr1("TestName");
```

```
sampleServiceV1Client.sampleOperation(v1Request);
```

Die Protokollierung auf Wire-Ebene zeigt, dass der `listParam` Parameter nicht serialisiert ist.

```
Action=SampleOperation&Version=2011-01-01&str1=TestName
```

Example von **SampleOperation** in 2.x

```
sampleServiceV2Client.sampleOperation(b -> b
    .str1("TestName"));
```

Die Protokollierung auf Wire-Ebene zeigt, dass der `listParam` Parameter ohne Wert serialisiert wird.

```
Action=SampleOperation&Version=2011-01-01&str1=TestName&listParam=
```

## Servicespezifische Änderungen

### Amazon S3-Änderungen

SDK for Java 2.x deaktiviert den anonymen Zugriff standardmäßig. Daher müssen Sie den anonymen Zugriff mithilfe der `aktivierenAnonymousCredentialsProvider`.

#### Änderungen des Vorgangsnamens

Viele der Operationsnamen für den Amazon S3 Client haben sich im AWS SDK for Java 2.x geändert. In Version 1.x wird der Amazon S3-Client nicht direkt aus der Service-API generiert. Dies führt zu Inkonsistenzen zwischen den SDK-Operationen und der Service-API. In Version 2.x wird der Amazon S3-Client jetzt konsistenter mit der Service-API erstellt.

Die folgende Tabelle zeigt die Operationsnamen in den beiden Versionen.

#### Amazon S3-Operationsnamen

1.x	2.x
<code>abortMultipartUpload</code>	<code>abortMultipartUpload</code>
<code>changeObjectStorageClass</code>	<code>copyObject</code>
<code>completeMultipartUpload</code>	<code>completeMultipartUpload</code>

1.x	2.x
<code>copyObject</code>	<code>copyObject</code>
<code>copyPart</code>	<code>uploadPartCopy</code>
<code>createBucket</code>	<code>createBucket</code>
<code>deleteBucket</code>	<code>deleteBucket</code>
<code>deleteBucketAnalyticsConfiguration</code>	<code>deleteBucketAnalyticsConfiguration</code>
<code>deleteBucketCrossOriginConfiguration</code>	<code>deleteBucketCors</code>
<code>deleteBucketEncryption</code>	<code>deleteBucketEncryption</code>
<code>deleteBucketInventoryConfiguration</code>	<code>deleteBucketInventoryConfiguration</code>
<code>deleteBucketLifecycleConfiguration</code>	<code>deleteBucketLifecycle</code>
<code>deleteBucketMetricsConfiguration</code>	<code>deleteBucketMetricsConfiguration</code>
<code>deleteBucketPolicy</code>	<code>deleteBucketPolicy</code>
<code>deleteBucketReplicationConfiguration</code>	<code>deleteBucketReplication</code>
<code>deleteBucketTaggingConfiguration</code>	<code>deleteBucketTagging</code>
<code>deleteBucketWebsiteConfiguration</code>	<code>deleteBucketWebsite</code>
<code>deleteObject</code>	<code>deleteObject</code>
<code>deleteObjectTagging</code>	<code>deleteObjectTagging</code>
<code>deleteObjects</code>	<code>deleteObjects</code>

1.x	2.x
<code>deleteVersion</code>	<code>deleteObject</code>
<code>disableRequesterPays</code>	<code>putBucketRequestPayment</code>
<code>doesBucketExist</code>	<code>headBucket</code>
<code>doesBucketExistV2</code>	<code>headBucket</code>
<code>doesObjectExist</code>	<code>headObject</code>
<code>enableRequesterPays</code>	<code>putBucketRequestPayment</code>
<code>generatePresignedUrl</code>	<a href="#">S3Presigner</a>
<code>getBucketAccelerateConfiguration</code>	<code>getBucketAccelerateConfiguration</code>
<code>getBucketAcl</code>	<code>getBucketAcl</code>
<code>getBucketAnalyticsConfiguration</code>	<code>getBucketAnalyticsConfiguration</code>
<code>getBucketCrossOriginConfiguration</code>	<code>getBucketCors</code>
<code>getBucketEncryption</code>	<code>getBucketEncryption</code>
<code>getBucketInventoryConfiguration</code>	<code>getBucketInventoryConfiguration</code>
<code>getBucketLifecycleConfiguration</code>	<code>getBucketLifecycle</code> oder <code>getBucketLifecycleConfiguration</code>
<code>getBucketLocation</code>	<code>getBucketLocation</code>
<code>getBucketLoggingConfiguration</code>	<code>getBucketLogging</code>
<code>getBucketMetricsConfiguration</code>	<code>getBucketMetricsConfiguration</code>
<code>getBucketNotificationConfiguration</code>	<code>getBucketNotification</code> oder <code>getBucketNotificationConfiguration</code>

1.x	2.x
<code>getBucketPolicy</code>	<code>getBucketPolicy</code>
<code>getBucketReplicationConfiguration</code>	<code>getBucketReplication</code>
<code>getBucketTaggingConfiguration</code>	<code>getBucketTagging</code>
<code>getBucketVersioningConfiguration</code>	<code>getBucketVersioning</code>
<code>getBucketWebsiteConfiguration</code>	<code>getBucketWebsite</code>
<code>getObject</code>	<code>getObject</code>
<code>getObjectAcl</code>	<code>getObjectAcl</code>
<code>getObjectAsString</code>	<code>getObjectAsBytes().asUtf8String</code>
<code>getObjectMetadata</code>	<code>headObject</code>
<code>getObjectTagging</code>	<code>getObjectTagging</code>
<code>getResourceUrl</code>	<a href="#">S3Utilities#getUrl</a>
<code>getS3AccountOwner</code>	<code>listBuckets</code>
<code>getUrl</code>	<a href="#">S3Utilities#getUrl</a>
<code>headBucket</code>	<code>headBucket</code>
<code>initiateMultipartUpload</code>	<code>createMultipartUpload</code>
<code>isRequesterPaysEnabled</code>	<code>getBucketRequestPayment</code>
<code>listBucketAnalyticsConfigurations</code>	<code>listBucketAnalyticsConfigurations</code>
<code>listBucketInventoryConfigurations</code>	<code>listBucketInventoryConfigurations</code>

1.x	2.x
<code>listBucketMetricsConfigurations</code>	<code>listBucketMetricsConfigurations</code>
<code>listBuckets</code>	<code>listBuckets</code>
<code>listMultipartUploads</code>	<code>listMultipartUploads</code>
<code>listNextBatchOfObjects</code>	<code>listObjectsV2Paginator</code>
<code>listNextBatchOfVersions</code>	<code>listObjectVersionsPaginator</code>
<code>listObjects</code>	<code>listObjects</code>
<code>listObjectsV2</code>	<code>listObjectsV2</code>
<code>listParts</code>	<code>listParts</code>
<code>listVersions</code>	<code>listObjectVersions</code>
<code>putObject</code>	<code>putObject</code>
<code>restoreObject</code>	<code>restoreObject</code>
<code>restoreObjectV2</code>	<code>restoreObject</code>
<code>selectObjectContent</code>	<code>selectObjectContent</code>
<code>setBucketAccelerateConfiguration</code>	<code>putBucketAccelerateConfiguration</code>
<code>setBucketAcl</code>	<code>putBucketAcl</code>
<code>setBucketAnalyticsConfiguration</code>	<code>putBucketAnalyticsConfiguration</code>
<code>setBucketCrossOriginConfiguration</code>	<code>putBucketCors</code>
<code>setBucketEncryption</code>	<code>putBucketEncryption</code>
<code>setBucketInventoryConfiguration</code>	<code>putBucketInventoryConfiguration</code>



1.x	2.x
setBucketLifecycleConfiguration	putBucketLifecycle oder putBucketLifecycleConfiguration
setBucketLoggingConfiguration	putBucketLogging
setBucketMetricsConfiguration	putBucketMetricsConfiguration
setBucketNotificationConfiguration	putBucketNotification oder putBucketNotificationConfiguration
setBucketPolicy	putBucketPolicy
setBucketReplicationConfiguration	putBucketReplication
setBucketTaggingConfiguration	putBucketTagging
setBucketVersioningConfiguration	putBucketVersioning
setBucketWebsiteConfiguration	putBucketWebsite
setObjectAcl	putObjectAcl
setObjectRedirectLocation	copyObject
setObjectTagging	putObjectTagging
uploadPart	uploadPart

## Amazon SNS-Änderungen

Ein SNS-Client kann nicht mehr auf SNS-Themen in anderen Regionen als der Region zugreifen, für die er konfiguriert ist.

## Amazon SQS-Änderungen

Ein SQS-Client kann nicht mehr auf SQS-Warteschlangen in anderen Regionen als der Region zugreifen, für die er konfiguriert ist.

## Amazon RDS-Änderungen

Das SDK for Java 2.x verwendet `RdsUtilities#generateAuthenticationToken` anstelle der Klasse `RdsIamAuthTokenGenerator` in 1.x.

## Änderungen der Profildatei

Der AWS SDK for Java 2.x analysiert die Profildefinitionen in `~/.aws/config` und `~/.aws/credentials` um die Art und Weise, wie die AWS CLI die Dateien analysiert, genauer zu emulieren.

Das SDK für Java 2.x:

- Löst ein `~/` oder `~` gefolgt vom Standardpfadtrennzeichen des Dateisystems zu Beginn des Pfads auf, indem Sie in der Reihenfolge `$HOME`, `$USERPROFILE` (nur Windows), `$HOMEDRIVE`, `$HOMEPATH` (nur Windows) und dann die `user.home` Systemeigenschaft überprüfen.
- Sucht nach der `AWS_SHARED_CREDENTIALS_FILE` Umgebungsvariablen anstelle von `AWS_CREDENTIAL_PROFILES_FILE`.
- Entfernt Profildefinitionen in Konfigurationsdateien ohne das Wort `profile` am Anfang des Profilnamens.
- Entfernt stillschweigend Profildefinitionen, die nicht aus alphanumerischen Zeichen, Unterstrichen oder Bindestrichen bestehen (nachdem das führende `profile` Wort für Konfigurationsdateien entfernt wurde).
- Führt Einstellungen von Profildefinitionen zusammen, die in derselben Datei dupliziert wurden.
- Führt Einstellungen von Profildefinitionen zusammen, die sowohl in der Konfigurations- als auch in der Anmeldeinformationsdatei dupliziert wurden.
- Führt die Einstellungen NICHT zusammen, wenn `[profile foo]` sich sowohl als auch in derselben Datei `[foo]` befinden.
- Verwendet Einstellungen in `[profile foo]`, wenn `[profile foo]` sowohl als auch in der Konfigurationsdatei gefunden `[foo]` werden.
- Verwendet den Wert der zuletzt duplizierten Einstellung in derselben Datei und demselben Profil.
- Erkennt `;` sowohl als auch `#` zum Definieren eines Kommentars.

- Erkennt ; und # in Profildefinitionen, um einen Kommentar zu definieren, auch wenn sich die Zeichen neben der schließenden Klammer befinden.
- Erkennt ; und , # um einen Kommentar nur beim Festlegen von Werten zu definieren, wenn ihnen Leerzeichen vorangestellt sind.
- Erkennt ; und # und alle folgenden Inhalte beim Festlegen von Werten, wenn ihnen keine Leerzeichen vorangestellt sind.
- Betrachtet rollenbasierte Anmeldeinformationen als Anmeldeinformationen mit der höchsten Priorität. Das 2.x SDK verwendet immer rollenbasierte Anmeldeinformationen, wenn der Benutzer die `-role_arn`Eigenschaft angibt.
- Erwägt sitzungsbasierte Anmeldeinformationen als second-highest-priority Anmeldeinformationen. Das 2.x SDK verwendet immer sitzungsbasierte Anmeldeinformationen, wenn rollenbasierte Anmeldeinformationen nicht verwendet wurden und der Benutzer die `aws_session_token` Eigenschaften `aws_access_key_id` und angibt.
- Verwendet grundlegende Anmeldeinformationen, wenn rollenbasierte und sitzungsbasierte Anmeldeinformationen nicht verwendet werden und der Benutzer die `aws_access_key_id` Eigenschaft angegeben hat.

## Änderungen an Umgebungsvariablen und Systemeigenschaften

1.x Umgebungsvariable	1.x Systemeigenschaft	2.x Umgebungsvariable	2.x Systemeigenschaft
AWS_ACCESS_KEY_ID AWS_ACCESS_KEY	<code>aws.accessKeyId</code>	AWS_ACCESS_KEY_ID	<code>aws.accessKeyId</code>
AWS_SECRET_KEY AWS_SECRET_ACCESS_KEY	<code>aws.secretKey</code>	AWS_SECRET_ACCESS_KEY	<code>aws.secretAccessKey</code>
AWS_SESSION_TOKEN	<code>aws.sessionToken</code>	AWS_SESSION_TOKEN	<code>aws.sessionToken</code>
AWS_REGION	<code>aws.region</code>	AWS_REGION	<code>aws.region</code>
AWS_CONFIG_FILE		AWS_CONFIG_FILE	<code>aws.configFile</code>

1.x Umgebungsvariable	1.x Systemeigenschaft	2.x Umgebungsvariable	2.x Systemeigenschaft
AWS_CREDENTIALS_PROFILE_FILE		AWS_SHARED_CREDENTIALS_FILE	<code>aws.sharedCredentialsFile</code>
AWS_PROFILE	<code>aws.profile</code>	AWS_PROFILE	<code>aws.profile</code>
AWS_EC2_METADATA_DISABLED	<code>com.amazonaws.sdk.disableEc2Metadata</code>	AWS_EC2_METADATA_DISABLED	<code>aws.disableEc2Metadata</code>
	<code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	AWS_EC2_METADATA_SERVICE_ENDPOINT	<code>aws.ec2MetadataServiceEndpoint</code>
AWS_CONTAINER_CREDENTIALS_RELATIVE_URI		AWS_CONTAINER_CREDENTIALS_RELATIVE_URI	<code>aws.containerCredentialsPath</code>
AWS_CONTAINER_CREDENTIALS_FULL_URI		AWS_CONTAINER_CREDENTIALS_FULL_URI	<code>aws.containerCredentialsFullUri</code>
AWS_CONTAINER_AUTHORIZATION_TOKEN		AWS_CONTAINER_AUTHORIZATION_TOKEN	<code>aws.containerAuthorizationToken</code>

1.x Umgebungsvariable	1.x Systemeigenschaft	2.x Umgebungsvariable	2.x Systemeigenschaft
AWS_CBOR_DISABLED	com.amazonaws.sdk.disableCbor	CBOR_ENABLED	aws.cborEnabled
AWS_ION_BINARY_DISABLE	com.amazonaws.sdk.disableIonBinary	BINARY_ION_ENABLED	aws.binaryIonEnabled
AWS_EXECUTION_ENV		AWS_EXECUTION_ENV	aws.executionEnvironment
	com.amazonaws.sdk.disableCertificateChecking	Nicht unterstützt ( <a href="#">Funktion anfordern</a> )	Nicht unterstützt ( <a href="#">Funktion anfordern</a> )
	com.amazonaws.sdk.enableDefaultMetrics	<a href="#">Nicht unterstützt</a>	<a href="#">Nicht unterstützt</a>
	com.amazonaws.sdk.enableThrottledRetry	<a href="#">Nicht unterstützt</a>	<a href="#">Nicht unterstützt</a>
	com.amazonaws.regions.RegionUtils.fileOverride	Nicht unterstützt ( <a href="#">Funktion anfordern</a> )	Nicht unterstützt ( <a href="#">Funktion anfordern</a> )

1.x Umgebungsvariable	1.x Systemeigenschaft	2.x Umgebungsvariable	2.x Systemeigenschaft
	<code>com.amazonaws.regions.RegionUtils.disableRemote</code>	Nicht unterstützt ( <a href="#">Funktion anfordern</a> )	Nicht unterstützt ( <a href="#">Funktion anfordern</a> )
	<code>com.amazonaws.services.s3.disableImplicitGlobalClients</code>	Nicht unterstützt ( <a href="#">Funktion anfordern</a> )	Nicht unterstützt ( <a href="#">Funktion anfordern</a> )
	<code>com.amazonaws.sdk.enableInRegionOptimizedMode</code>	Nicht unterstützt ( <a href="#">Funktion anfordern</a> )	Nicht unterstützt ( <a href="#">Funktion anfordern</a> )

## Änderungen in Waiters von Version 1 zu Version 2

In diesem Thema werden die Änderungen der Funktionalität von Waiters von Version 1 (v1) zu Version 2 (v2) beschrieben.

Die folgenden Tabellen zeigen speziell den Unterschied für DynamoDB-Warter. Waiter für andere Services folgen demselben Muster.

### Änderungen auf hoher Ebene

Waiter-Klassen befinden sich im selben Maven-Artefakt wie der Service.

Änderung	v1	v2
Maven-Abhängigkeiten	<code>&lt;dependencyManagement&gt;</code>	<code>&lt;dependencyManagement&gt;</code>

Änderung	v1	v2
	<pre> &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;     &lt;artifact Id&gt;aws-java-sdk-bom&lt;/ artifactId&gt;     &lt;version&gt; 1.12.680<sup>1</sup>&lt;/version&gt;     &lt;type&gt;pom&lt;/ type&gt;     &lt;scope&gt;im port&lt;/scope&gt;   &lt;/dependency&gt; &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;     &lt;artifact Id&gt;dynamodb&lt;/artif actId&gt;   &lt;/dependency&gt; &lt;/dependencies&gt; </pre>	<pre> &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;     &lt;artifact Id&gt;bom&lt;/artifactId&gt;     &lt;version&gt; 2.25.10<sup>2</sup>&lt;/version&gt;     &lt;type&gt;pom&lt;/ type&gt;     &lt;scope&gt;im port&lt;/scope&gt;   &lt;/dependency&gt; &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;     &lt;artifact Id&gt;dynamodb&lt;/artif actId&gt;   &lt;/dependency&gt; &lt;/dependencies&gt; </pre>
Package name	com.amazonaws.services.dynamodbv2.waiters	software.amazon.awssdk.services.dynamodb.waiters
Klassennamen	<a href="#">AmazonDynamoDBWaiters</a>	<ul style="list-style-type: none"> <li>• Synchron: <a href="#">DynamoDbWaiter</a></li> <li>• Asynchron: <a href="#">DynamoDbAsyncWaiter</a></li> </ul>

<sup>1</sup> [Neueste Version](#). <sup>2</sup> [Neueste Version](#).

## API-Änderungen

Änderung	v1	v2
Erstellen eines Waiters	<pre>AmazonDynamoDB client     = AmazonDynamoDBClientBuilder          .standard().build(     ); AmazonDynamoDBWaiters     waiter = client.waiters();</pre>	<p><b>Synchron:</b></p> <pre>DynamoDbClient client     = DynamoDbClient.create(); DynamoDbWaiter waiter     = client.waiter();</pre> <p><b>Asynchron:</b></p> <pre>DynamoDbAsyncClient     asyncClient =         DynamoDbAsyncClient.create(); DynamoDbAsyncWaiter     waiter = asyncClient.waiter();</pre>
Warten Sie, bis eine Tabelle vorhanden ist	<p><b>Synchron:</b></p> <pre>waiter.tableExists()     .run(new WaiterParameters&lt;&gt;(         new DescribeTableRequest(tableName)));</pre> <p><b>Asynchron:</b></p> <pre>waiter.tableExists()     .runAsync(new WaiterParameters()         .withRequest(new DescribeTableRequest(tableName)),</pre>	<p><b>Synchron:</b></p> <pre>WaiterResponse&lt;DescribeTableResponse&gt;     waiterResponse =         waiter.waitUntilTableExists(             r -&gt; r.tableName("myTable")); waiterResponse.matched().response()     .ifPresent(System.out::println);</pre> <p><b>Asynchron:</b></p>



Änderung	v1	v2
	<pre>                 new WaiterHan dler() {                     @Override                     public                     void onWaitSuccess(                      AmazonWebServiceRe quest amazonWeb ServiceRequest) {                          System.out.println ("Table creation succeeded");                     }                      @Override                     public                     void onWaitFai lure(Exception e) {                          e.printStackTrace();                     }                 }).get();             </pre>	<pre> waiter.waitUntilTa bleExists(r -&gt;     r.tableName(tableN ame))                 .whenComp lete((r, t) -&gt; {                     if (t != null) {                          t.printStackTrace();                     } else {                          System.out.println(                              "Table creation succeeded");                         }                 }).join();             </pre>

## Konfigurationsänderungen

Änderung	v1	v2
Abfragestrategie (max. Versuche und feste Verzögerung)	<pre> MaxAttemptsRetrySt rategy maxAttemp tsRetryStrategy =                 new MaxAttemp tsRetryStrategy(10);  FixedDelayStrategy fixedDelayStrategy =                 new FixedDela yStrategy(3);             </pre>	<pre> FixedDelayBackoff Strategy fixedDela yBackoffStrategy =                 FixedDela yBackoffStrategy                 .create(D uration.ofSeconds( 3));             </pre>

Änderung	v1	v2
	<pre> PollingStrategy     pollingStrategy =         new PollingStrategy(maxAttempts RetryStrategy,      fixedDelayStrategy);  waiter.tableExists().run(     new WaiterParameters&lt;&gt;(         new         DescribeTableRequest(tableName)),         pollingStrategy); </pre>	<pre> waiter.waitUntilTableExists(r -&gt; r.tableName(tableName),     c -&gt; c.maxAttempts(10)      .backoffStrategy(fixedDelayBackoffStrategy)); </pre>

## Änderungen in Amazon S3 Transfer Manager von Version 1 zu Version 2

In diesem Thema werden die Änderungen im Amazon S3 Transfer Manager von Version 1 (v1) zu Version 2 (v2) beschrieben.

### Änderungen auf hoher Ebene

Änderung	v1	v2
Maven-Abhängigkeiten	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; com.amazonaws&lt;/groupId&gt;       &lt;artifactId&gt;aws-java-sdk-bom&lt;/artifactId&gt;       &lt;version&gt; 1.12.587&lt;/version&gt; </pre>	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; software.amazon.awssdk&lt;/groupId&gt;       &lt;artifactId&gt;bom&lt;/artifactId&gt;       &lt;version&gt; 2.21.21&lt;/version&gt;       &lt;type&gt;pom&lt;/type&gt; </pre>

Änderung	v1	v2
	<pre>                     &lt;type&gt;pom&lt;/ type&gt;                     &lt;scope&gt;im port&lt;/scope&gt;                     &lt;/dependency&gt;                     &lt;/dependencies&gt; &lt;/dependencyManagem ent&gt; &lt;dependencies&gt;     &lt;dependency&gt;         &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;         &lt;artifact Id&gt;aws-java-sdk-s3&lt;/ artifactId&gt;         &lt;/dependency&gt;     &lt;/dependencies&gt;                 </pre>	<pre>                     &lt;scope&gt;im port&lt;/scope&gt;                     &lt;/dependency&gt;                     &lt;/dependencies&gt; &lt;/dependencyManagem ent&gt; &lt;dependencies&gt;     &lt;dependency&gt;         &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;         &lt;artifactId&gt;s3- transfer-manager&lt;/art ifactId&gt;         &lt;/dependency&gt;     &lt;dependency&gt;         &lt;groupId&gt; software.amazon.aw ssdk.crt&lt;/groupId&gt;         &lt;artifa ctId&gt;aws-crt&lt;/artifa ctId&gt;         &lt;version&gt; 0.28.7<sup>3</sup>&lt;/version&gt;         &lt;/dependency&gt;     &lt;/dependencies&gt;                 </pre>
Package name	com.amazonaws.serv ices.s3.transfer	software.amazon.aw ssdk.transfer.s3
Class name	<a href="#">TransferManager</a>	<a href="#">S3TransferManager</a>

<sup>1</sup> [Neueste Version](#) . <sup>2</sup> [Neueste Version](#) . <sup>3</sup> [Neueste Version](#) .

## Konfigurations-API-Änderungen

Einstellung	v1	v2
(einen Builder abrufen)	<pre>TransferManagerBuilder tmBuilder =     TransferManagerBui lder.standard();</pre>	<pre>S3TransferManager. Builder tmBuilder =     S3TransferManager. builder();</pre>
S3-Client	<pre>tmBuilder.withS3Cl ient(...); tmBuilder.setS3C lient(...);</pre>	<pre>tmBuilder.s3Client (...);</pre>
Executor	<pre>tmBuilder.withExec utorFactory(...); tmBuilder.setExecu torFactory(...);</pre>	<pre>tmBuilder.executor (...);</pre>
Herunterfahren von Threadpools	<pre>tmBuilder.withShut DownThreadPools(...); tmBuilder.setS hutdownThreadPools (...);</pre>	<p>Nicht unterstützt Der bereitges tellte Executor wird nicht heruntergefahren, wenn der S3TransferManager geschloss en wird</p>
Minimale Größe des Upload-Teils	<pre>tmBuilder.withMini mumUploadPartSize( ...); tmBuilder.setMinimumU ploadPartSize(...);</pre>	<pre>S3AsyncClient s3 =     S3AsyncClient.crtB uilder().     minimumPa rtSizeInBytes(...) .build();  tmBuilder.s3Clie nt(s3);</pre>
Schwellenwert für mehrteilige Uploads	<pre>tmBuilder.withMini mumUploadPartSize( ...);</pre>	<pre>S3AsyncClient s3 =     S3AsyncClient.crtB uilder().</pre>

Einstellung	v1	v2
	<pre>tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>threshold InBytes(...).build();  tmBuilder.s3Client(s3) ;</pre>
Minimale Größe des Kopierteils	<pre>tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 =     S3AsyncClient.crtBuilder()         .minimumPartSizeInBytes(...)         .build();  tmBuilder.s3Client(s3);</pre>
Schwellenwert für mehrteilige Kopien	<pre>tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 =     S3AsyncClient.crtBuilder()         .threshold         InBytes(...).build();  tmBuilder.s3Client(s3) ;</pre>
Deaktivieren paralleler Downloads	<pre>tmBuilder.withDisableParallelDownloads(...); tmBuilder.setDisableParallelDownloads(...);</pre>	<p>Deaktivieren Sie parallele Downloads, indem Sie einen standardmäßigen Java-basierten S3-Client an den Transfer Manager übergeben.</p> <pre>S3AsyncClient s3 =     S3AsyncClient.builder().build();  tmBuilder.s3Client(s3);</pre>

Einstellung	v1	v2
Immer mehrteilige md5 berechnen	<pre>tmBuilder.withAlwaysCalculateMultiPartMd5(...); tmBuilder.setAlwaysCalculateMultiPartMd5(...);</pre>	Nicht unterstützt

## Verhaltensänderungen



Parallele Übertragung erfordert einen AWS CRT-basierten S3-Client



Im SDK für Java 2.x ist die automatische parallele Übertragung (mehnteiliges Hochladen/ Herunterladen) über den [AWS CRT-basierten S3-Client](#) verfügbar. Um das Feature für parallele Übertragungen zu aktivieren, müssen Sie die [AWS Common Runtime \(CRT\)-Bibliotheksabhängigkeit](#) explizit hinzufügen, um die maximale Leistung zu erzielen.



Der AWS CRT-basierte S3-Client allein – ohne Verwendung von `S3TransferManager` – bietet eine maximale Leistung paralleler Übertragungen. `S3TransferManager v2` bietet zusätzliche APIs, die die Übertragung von Dateien und Verzeichnissen erleichtern.

Die Fähigkeit des `S3TransferManager`, parallele Übertragungen durchzuführen, hängt davon ab, wie initiiert `S3TransferManager` wird und ob die AWS Common Runtime (CRT)-Bibliothek als Abhängigkeit deklariert wurde.

In der folgenden Tabelle werden drei Initialisierungsszenarien für einen `S3TransferManager v2` mit und ohne das AWS CRT beschrieben, das als Abhängigkeit deklariert ist.

S3TransferManager v2-Initialisierungsansatz	Ist AWS CRT als Abhängigkeit deklariert?	
	Ja	Nein
Initialisieren der <b>S3TransferManager</b> ohne Übergabe einer <b>S3AsyncClient</b> Instance  Statische Erstellungsmethode:		

S3TransferManager v2-Initialisierungsansatz	Ist AWS CRT als Abhängigkeit deklariert?	
<p><code>S3TransferManager.create();</code></p> <p>- ODER -</p> <p>Builder-Methode:</p> <p><code>S3TransferManager.builder().build();</code></p>	<p>Automatische parallele Übertragung aktiviert</p>	<p>Automatische parallele Übertragung deaktiviert</p>
<p>Übergeben einer <b>S3AsyncClient</b> Instance, die mit einer der crt*(-)Builder-Methoden erstellt wurde</p> <p><code>S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder().build();</code>  <code>S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</code></p> <p>- ODER -</p> <p><code>S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate();</code>  <code>S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</code></p>	<p></p> <p>Automatische parallele Übertragung aktiviert</p>	<p></p> <p>Laufzeitfehler</p>

S3TransferManager v2-Initialisierungsansatz	Ist AWS CRT als Abhängigkeit deklariert?	
<p>Übergeben Sie eine <b>S3AsyncClient</b> Instance, die mit einer der Standard-Builder-Methoden erstellt wurde, sodass der Transfer Manager keinen Verweis auf das CRT hat</p> <pre data-bbox="126 443 1019 642">S3AsyncClient s3AsyncClient = S3AsyncClient.builder().build(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre> <p>- ODER -</p> <pre data-bbox="126 751 1019 951">S3AsyncClient s3AsyncClient = S3AsyncClient.create(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre>	<div data-bbox="1094 279 1222 390" style="text-align: center;"></div> <p data-bbox="1065 443 1250 615">Automatische parallele Übertragung deaktiviert</p>	<div data-bbox="1338 279 1466 390" style="text-align: center;"></div> <p data-bbox="1308 443 1494 615">Automatische parallele Übertragung deaktiviert</p>

## Paralleler Download über Abrufe im Bytebereich

Wenn die automatische parallele Übertragung aktiviert ist, verwendet S3 Transfer Manager v2 [Abrufe im Bytebereich](#), um bestimmte Teile des Objekts parallel abzurufen (mehrteiliger Download). Die Art und Weise, wie ein Objekt mit v2 heruntergeladen wird, hängt nicht davon ab, wie das Objekt ursprünglich hochgeladen wurde. Alle Downloads können von hohem Durchsatz und Gleichzeitigkeit profitieren.

Im Gegensatz dazu ist es bei S3 Transfer Manager v1 wichtig, wie das Objekt ursprünglich hochgeladen wurde. S3 Transfer Manager v1 ruft die Teile des Objekts auf die gleiche Weise ab, wie die Teile hochgeladen wurden. Wenn ein Objekt ursprünglich als einzelnes Objekt hochgeladen wurde, kann S3 Transfer Manager v1 den Download-Prozess nicht mithilfe von Unteranforderungen beschleunigen.

## Fehlerverhalten

Mit S3 Transfer Manager v1 schlägt eine Verzeichnisübertragungsanforderung fehl, wenn eine Unteranforderung fehlschlägt. Im Gegensatz zu v1 wird die von S3 Transfer Manager v2



zurückgegebene Zukunft erfolgreich abgeschlossen, auch wenn einige Unteranforderungen fehlschlagen.

Daher sollten Sie die Antwort auf Fehler überprüfen, indem Sie die [-CompletedDirectoryDownload.failedTransfers\(\)](#) Methode oder [-CompletedDirectoryUpload.failedTransfers\(\)](#) Methode verwenden, auch wenn die Zukunft erfolgreich abgeschlossen wird.

## Änderungen im EC2-Metadaten-Dienstprogramm von Version 1 zu Version 2

In diesem Thema werden die Änderungen im Metadaten-Dienstprogramm SDK for Java Amazon Elastic Compute Cloud (EC2) von Version 1 (v1) zu Version 2 (v2) beschrieben.

### Allgemeine Änderungen

Änderung	v1	v2
Maven-Abhängigkeiten	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;       &lt;artifact Id&gt;aws-java-sdk-bom&lt;/ artifactId&gt;       &lt;version&gt; 1.12.587&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt; </pre>	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;       &lt;artifact Id&gt;bom&lt;/artifactId&gt;       &lt;version&gt; 2.21.21&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt; </pre>

Änderung	v1	v2
	<pre data-bbox="609 210 1006 541"> &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;     &lt;artifact Id&gt;aws-java-sdk-co re&lt;/artifactId&gt;     &lt;/dependency&gt; &lt;/dependencies&gt; </pre>	<pre data-bbox="1084 210 1482 699"> &lt;artifact Id&gt;imds&lt;/artifactId&gt; &lt;/dependency&gt; &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;     &lt;artifact Id&gt;apache-client<sup>3</sup>&lt;/ artifactId&gt;     &lt;/dependency&gt; &lt;/dependencies&gt; </pre>
Package name	com.amazonaws.util	software.amazon.awssdk.imds
Instanziierungsansatz	<p data-bbox="592 871 982 997">Verwenden Sie statische Dienstprogrammmethoden; keine Instanziierung:</p> <pre data-bbox="609 1060 1006 1228"> String localHostName =     EC2Metada taUtils.getLocalHo stName(); </pre>	<p data-bbox="1068 871 1485 955">Verwenden Sie eine statische Factory-Methode:</p> <pre data-bbox="1084 1018 1482 1144"> Ec2MetadataClient client = Ec2Metada taClient.create(); </pre> <p data-bbox="1068 1186 1485 1270">Oder verwenden Sie einen Builder-Ansatz:</p> <pre data-bbox="1084 1333 1482 1564"> Ec2MetadataClient client = Ec2Metada taClient.builder()     .endpointMode(Endp ointMode.IPV6)     .build(); </pre>
Client-Typen	Synchrone Dienstprogrammmethoden: EC2MetadataUtils	Synchron: Ec2MetadataClient Asynchron: Ec2MetadataAsyncClient

<sup>1</sup> [Neueste Version](#) . <sup>2</sup> [Neueste Version](#) .

<sup>3</sup> Beachten Sie die Deklaration des `apache-client` Moduls für v2. V2 des EC2-Metadaten-Hilfsprogramms erfordert eine Implementierung der `-SdkHttpClientSchnittstelle` für den synchronen Metadaten-Client oder der `-SdkAsyncHttpClientSchnittstelle` für den asynchronen Metadaten-Client. Der [???](#) Abschnitt zeigt die Liste der HTTP-Clients, die Sie verwenden können.

## Anfordern von Metadaten

In v1 verwenden Sie statische Methoden, die keine Parameter akzeptieren, um Metadaten für eine EC2-Ressource anzufordern. Im Gegensatz dazu müssen Sie den Pfad zur EC2-Ressource als Parameter in v2 angeben. Die folgende Tabelle zeigt die verschiedenen Ansätze.

v1	v2
<pre>String userMetaData = EC2Metada taUtils.getUserData();</pre>	<pre>Ec2MetadadataClient client = Ec2Metada taClient.create(); Ec2MetadadataResponse response =     client.get("/latest/ user-data"); String userMetaData =     response.asString();</pre>

In den [Metadatenkategorien der Instance](#) finden Sie den Pfad, den Sie angeben müssen, um Metadaten anzufordern.

### Note

Wenn Sie einen Instance-Metadaten-Client in v2 verwenden, sollten Sie versuchen, für alle Anfragen zum Abrufen von Metadaten denselben Client zu verwenden.

## Verhaltensänderungen

### JSON-Daten

Auf EC2 gibt der lokal ausgeführte Instance Metadata Service (IMDS) einige Metadaten als JSON-formatierte Zeichenfolgen zurück. Ein solches Beispiel sind die dynamischen Metadaten eines [Instance-Identitätsdokuments](#) .

Die v1-API enthält separate Methoden für jeden Teil von Instance-Identitätsmetadaten, wohingegen die v2-API die JSON-Zeichenfolge direkt zurückgibt. Um mit der JSON-Zeichenfolge zu arbeiten, können Sie die [Document API](#) verwenden, um die Antwort zu analysieren und in der JSON-Struktur zu navigieren.

In der folgenden Tabelle wird verglichen, wie Sie Metadaten eines Instance-Identitätsdokuments in v1 und v2 abrufen.

Anwendungsfall	v1	v2
Abrufen der Region	<pre>InstanceInfo instanceInfo =     EC2MetadataUtils.getInstanceInfo(); String region =     instanceInfo.getRegion();</pre>	<pre>Ec2MetadataResponse response =     client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String region =     instanceInfo.asMap().get("region").asString();</pre>
Abrufen der Instance-ID	<pre>InstanceInfo instanceInfo =     EC2MetadataUtils.getInstanceInfo(); String instanceId =     instanceInfo.getInstanceId();</pre>	<pre>Ec2MetadataResponse response =     client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String instanceId =     instanceInfo.asMap</pre>

Anwendungsfall	v1	v2
		<pre>().get("instanceId").asString();</pre>
Abrufen des Instance-Typs	<pre>InstanceInfo instanceInfo =     EC2MetadataUtils.getInstanceInfo(); String instanceType = instanceInfo.getInstanceType();</pre>	<pre>Ec2MetadataResponse response =     client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String instanceType = instanceInfo.asMap().get("instanceType").asString();</pre>

### Unterschiede bei der Endpunktauflösung

Die folgende Tabelle zeigt die Speicherorte, die das SDK prüft, um den Endpunkt in IMDS aufzulösen. Die Standorte werden in absteigender Priorität aufgeführt.

v1	v2
Systemeigenschaft: <code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	Client-Builder-Konfigurationsmethode: <code>endpoint(...)</code>
Umgebungsvariable: <code>AWS_EC2_METADATA_SERVICE_ENDPOINT</code>	Systemeigenschaft: <code>aws.ec2MetadataServiceEndpoint</code>
Standardwert: <code>http://169.254.169.254</code>	Config-Datei: <code>~.aws/config</code> mit der <code>ec2_metadata_service_endpoint</code> Einstellung
	Wert im Zusammenhang mit aufgelösten <code>endpoint-mode</code>

v1	v2
	Standardwert: <code>http://169.254.169.254</code>

## Endpunktauflösung in v2

Wenn Sie explizit einen Endpunkt mithilfe des Builders festlegen, hat dieser Endpunktwert Vorrang vor allen anderen Einstellungen. Wenn der folgende Code ausgeführt wird, werden die `aws.ec2MetadataServiceEndpoint` Systemeigenschaft und die `ec2_metadata_service_endpoint` Konfigurationsdateieinstellung ignoriert, wenn sie vorhanden sind.

```
Ec2MetadataClient client = Ec2MetadataClient
    .builder()
    .endpoint(URI.create("endpoint.to.use"))
    .build();
```

## Endpunktmodus

Mit v2 können Sie einen Endpunktmodus angeben, um den Metadaten-Client so zu konfigurieren, dass er die Standardendpunktwerte für IPv4 oder IPv6 verwendet. Der Endpunktmodus ist für v1 nicht verfügbar. Der für IPv4 verwendete Standardwert ist `http://169.254.169.254` und `http://[fd00:ec2::254]` für IPv6.

Die folgende Tabelle zeigt die verschiedenen Möglichkeiten, wie Sie den Endpunktmodus in der Reihenfolge der absteigenden Priorität festlegen können.

		Mögliche Werte
Client-Builder-Konfiguration methode: <code>endpointMode(...)</code>	<pre>Ec2MetadataClient     client = Ec2MetadataClient         .builder()         .endpointMode(EndpointMode.IPV4)         .build();</pre>	<code>EndpointMode.IPV4</code> , <code>EndpointMode.IPV6</code>

		Mögliche Werte
Systemeigenschaft	<code>aws.ec2MetadataServiceEndpointMode</code>	IPv4, IPv6 (Groß-/Kl einschreibung spielt keine Rolle)
Konfigurationsdatei: <code>~.aws/config</code>	<code>ec2_metadata_service_endpoint</code> -Einstellung	IPv4, IPv6 (Groß-/Kl einschreibung spielt keine Rolle)
In den vorherigen Methoden nicht angegeben	IPv4 wird verwendet	

Wie das SDK **endpoint** oder **endpoint-mode** in v2 auflöst

- Das SDK verwendet den Wert, den Sie im Code im Client Builder festgelegt haben, und ignoriert alle externen Einstellungen. Da das SDK eine Ausnahme auslöst, wenn `endpoint` sowohl als auch im Client Builder aufgerufen `endpointMode` werden, verwendet das SDK den Endpunktwert der verwendeten Methode.
- Wenn Sie keinen Wert im Code festlegen, sucht das SDK nach einer externen Konfiguration – zuerst nach Systemeigenschaften und dann nach einer Einstellung in der Konfigurationsdatei.
  - Das SDK prüft zunächst auf einen Endpunktwert. Wenn ein Wert gefunden wird, wird er verwendet.
  - Wenn das SDK immer noch keinen Wert gefunden hat, sucht das SDK nach Einstellungen für den Endpunktmodus.
- Wenn das SDK keine externen Einstellungen findet und Sie den Metadaten-Client nicht im Code konfiguriert haben, verwendet das SDK schließlich den IPv4-Wert von `http://169.254.169.254`.

## IMDSv2

Amazon EC2 definiert zwei Ansätze für den Zugriff auf Instance-Metadaten:

- Instance Metadata Service Version 1 (IMDSv1) – Anfrage-/Antwortansatz
- Instance Metadata Service Version 2 (IMDSv2) – Sitzungsorientierter Ansatz

In der folgenden Tabelle wird verglichen, wie die Java SDKs mit IMDS funktionieren.

v1	v2
IMDSv2 wird standardmäßig verwendet	Verwendet immer IMDSv2
Versucht, ein Sitzungstoken für jede Anforderung abzurufen, und greift auf IMDSv1 zurück, wenn es ein Sitzungstoken nicht abrufen kann	Behält ein Sitzungstoken in einem internen Cache, der für mehrere Anforderungen wiederverwendet wird

Das SDK for Java 2.x unterstützt nur IMDSv2 und greift nicht auf IMDSv1 zurück.

## Konfigurationsunterschiede

In der folgenden Tabelle sind die unterschiedlichen Konfigurationsoptionen aufgeführt.

Konfiguration	v1	v2
Wiederholversuche	Konfiguration nicht verfügbar	Über Builder-Methode konfigurierbar <code>retryPolicy(...)</code>
HTTP	Verbindungs-Timeout, das über die <code>AWS_METADATA_SERVICE_TIMEOUT</code> Umgebungsvariable konfiguriert werden kann. Der Standardwert ist 1 Sekunde.	Die Konfiguration ist verfügbar, indem ein HTTP-Client an die Builder-Methode übergeben wird <code>httpClient(...)</code> . Das Standard-Verbindungs-Timeout für HTTP-Clients beträgt 2 Sekunden.

## Beispiel für eine v2-HTTP-Konfiguration

Das folgende Beispiel zeigt, wie Sie den Metadaten-Client konfigurieren können. In diesem Beispiel wird das Verbindungs-Timeout konfiguriert und der Apache-HTTP-Client verwendet.

```
SdkHttpClient httpClient = ApacheHttpClient.builder()
    .connectionTimeout(Duration.ofSeconds(1))
    .build();
```



```
Ec2MetadataClient imdsClient = Ec2MetadataClient.builder()
    .httpClient(httpClient)
    .build();
```

## Änderungen bei der Amazon- CloudFront Vorsignierung von Version 1 zu Version 2

In diesem Thema werden die Änderungen in Amazon CloudFront von Version 1 (v1) zu Version 2 (v2) beschrieben.

### Änderungen auf hoher Ebene

Änderung	v1	v2
Maven-Abhängigkeiten	<pre>&lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;       &lt;artifact Id&gt;aws-java-sdk-bom&lt;/ artifactId&gt;       &lt;version&gt; <b>1.12.587<sup>1</sup></b>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;</pre>	<pre>&lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;       &lt;artifact Id&gt;bom&lt;/artifactId&gt;       &lt;version&gt; <b>2.21.21<sup>2</sup></b>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;     &lt;artifact Id&gt;cloudfront&lt;/art ifactId&gt;</pre>

Änderung	v1	v2
	<pre>&lt;artifact Id&gt;cloudfront&lt;/art ifactId&gt; &lt;/dependency&gt; &lt;/dependencies&gt;</pre>	<pre>&lt;/dependency&gt; &lt;/dependencies&gt;</pre>
Package name	com.amazonaws.serv ices.cloudfront	software.amazon.aw ssdk.services.clou dfront
Klassennamen	<a href="#">CloudFrontUrlSigner</a>  <a href="#">CloudFrontCookieSigner</a>	<a href="#">CloudFrontUtilities</a>  <a href="#">SignedUrl</a>  <a href="#">CannedSignerRequest</a>  <a href="#">CustomSignerRequest</a>

<sup>1</sup> [Neueste Version](#) . <sup>2</sup> [Neueste Version](#) .

## API-Änderungen

Behavior	v1	v2
Erstellen einer vordefinierten Anforderung	Argumente werden direkt an die API übergeben.	<pre>CannedSignerRequest cannedRequest =     CannedSig nerRequest.builder()      .resourceUrl(resou rceUrl)      .privateKey(privat eKey)      .keyPairId(keyPairId)      .expirationDate(ex pirationDate)</pre>

Behavior	v1	v2
		<pre>.build();</pre>
Erstellen einer benutzerdefinierten Anforderung	Argumente werden direkt an die API übergeben.	<pre>CustomSignerRequest customRequest =     CustomSignerRequest.builder()          .resourceUrl(resourceUrl)          .privateKey(keyFile)          .keyPairId(keyPairId)          .expirationDate(expirationDate)          .activeDate(activeDate)          .ipRange(ipRange)          .build();</pre>
Generieren einer signierten URL (vordefiniert)	<pre>String signedUrl =     CloudFrontUrlSigner.getSignedURLWithCannedPolicy(         resourceUrl,         keyPairId, privateKey, expirationDate);</pre>	<pre>CloudFrontUtilities cloudFrontUtilities =     CloudFrontUtilities.create();  SignedUrl signedUrl =      cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedRequest);  String url = signedUrl.url();</pre>

Behavior	v1	v2
Generieren eines signierten Cookies (benutzerdefiniert)	<pre> CookiesForCustomPolicy cookies =     CloudFrontCookieSi gner.getCookiesFor CustomPolicy(     resourceUrl,     privateKey, keyPairId , expirationDate,     activeDate,     ipRange); </pre>	<pre> CloudFrontUtilities cloudFrontUtilities =     CloudFrontUtilitie s.create();  CookiesForCustomPolicy cookies =     cloudFrontUtilitie s.getCookiesForCus tomPolicy(customRe quest); </pre>

### Umgestaltete Cookie-Header in v2

In Java v1 stellt das Java SDK Cookie-Header als `bereitMap.Entry<String, String>`.

```

Map.Entry<String, String> signatureMap = cookies.getSignature();
String signatureKey = signatureMap.getKey(); // "CloudFront-Signature"
String signatureValue = signatureMap.getValue(); // "[SIGNATURE_VALUE]"

```

Das Java v2 SDK stellt den gesamten Header als einzelnes `bereitString`.

```

String signatureHeaderValue = cookies.signatureHeaderValue(); // "CloudFront-
Signature=[SIGNATURE_VALUE]"

```

## Änderungen beim Parsen von Amazon S3-URIs von Version 1 zu Version 2

In diesem Thema werden die Änderungen beim Parsen von Amazon S3-URIs von Version 1 (v1) zu Version 2 (v2.) beschrieben.

### Änderungen auf hoher Ebene

Um mit der Analyse eines S3-URI in v1 zu beginnen, instanzieren Sie einen `AmazonS3URI` mithilfe eines Konstruktors. In v2 rufen Sie `parseUri()` auf einer Instance von `aufS3Utilities`, um eine zurückzugegebenS3URI.

Änderung	v1	v2
Maven-Abhängigkeiten	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;       &lt;artifact Id&gt;aws-java-sdk-bom&lt;/ artifactId&gt;       &lt;version&gt; 1.12.587<sup>1</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;     &lt;artifact Id&gt;s3&lt;/artifactId&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; </pre>	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;       &lt;artifact Id&gt;bom&lt;/artifactId&gt;       &lt;version&gt; 2.21.21<sup>2</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;     &lt;artifact Id&gt;s3&lt;/artifactId&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; </pre>
Package name	com.amazonaws.serv ices.s3	software.amazon.aw ssdk.services.s3
Klassennamen	<a href="#">AmazonS3URI</a>	<a href="#">S3URI</a>

<sup>1</sup> [Neueste Version](#) . <sup>2</sup> [Neueste Version](#) .

## API-Änderungen

Behavior	v1	v2
Parsen Sie einen S3-URI.	<pre>URI uri = URI.create( "https://s3.amazonaws.com");  AmazonS3Uri s3Uri =     new AmazonS3URI(uri, false);</pre>	<pre>S3Client s3Client =     S3Client.create(); S3Utilities s3Utilities =     s3Client.utilities();  S3Uri s3Uri =     s3Utilities.parseUri(uri);</pre>
Rufen Sie den Bucket-Namen aus einem S3-URI ab.	<pre>String bucket =     s3Uri.getBucket();</pre>	<pre>Optional&lt;String&gt; bucket =     s3Uri.bucket();</pre>
Rufen Sie den Schlüssel ab.	<pre>String key = s3Uri.getKey();</pre>	<pre>Optional&lt;String&gt; key =     s3Uri.key();</pre>
Rufen Sie die Region ab.	<pre>String region =     s3Uri.getRegion();</pre>	<pre>Optional&lt;Region&gt; region =     s3Uri.region();  String region; if (s3Uri.region().isPresent()) {     region = s3Uri.region().get().id(); }</pre>
Rufen Sie ab, ob der S3-URI den Pfadstil hat.	<pre>boolean isPathStyle =     s3Uri.isPathStyle();</pre>	<pre>boolean isPathStyle =     s3Uri.isPathStyle();</pre>
Rufen Sie die Versions-ID ab.	<pre>String versionId =     s3Uri.getVersionId();</pre>	<pre>Optional&lt;String&gt;     versionId =</pre>

Behavior	v1	v2
		<pre>s3Uri.firstMatchin gRawQueryParameter ("versionId");</pre>
Rufen Sie die Abfragepa rameter ab.	N/A	<pre>Map&lt;String, List&lt;Stri ng&gt;&gt; queryParams =     s3Uri.rawQueryPara meters();</pre>

## Verhaltensänderungen

### URL-Kodierung

v1 bietet die Möglichkeit, ein `-Flag` zu übergeben, um anzugeben, ob der URI URL-kodiert werden soll. Der Standardwert ist `true`.

In v2 wird die URL-Kodierung nicht unterstützt. Wenn Sie mit Objektschlüsseln oder Abfrageparametern arbeiten, die reservierte oder unsichere Zeichen enthalten, müssen Sie sie per URL codieren. Sie müssen beispielsweise ein Leerzeichen durch ersetzen" `"%20`.

## Verwenden Sie das SDK for Java 1.x und 2.x side-by-side

Sie können beide AWS SDK for Java-Versionen in Ihren Projekten verwenden.

Im Folgenden wird ein Beispiel für die `pom.xml` Datei für ein Projekt gezeigt, das Versionen Amazon S3 ab Version 1.x und DynamoDB ab Version 2.16.1 verwendet.

### Example POM-Beispiel

Dieses Beispiel zeigt einen `pom.xml` Dateieintrag für ein Projekt, das sowohl 1.x- als auch 2.x-Versionen des SDK verwendet.

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-bom</artifactId>
```

```
        <version>1.12.1</version>
        <type>pom</type>
        <scope>import</scope>
    </dependency>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.16.1</version>
        <type>pom</type>
        <scope>import</scope>
    </dependency>
</dependencies>
</dependencyManagement>

<dependencies>
    <dependency>
        <groupId>com.amazonaws</groupId>
        <artifactId>aws-java-sdk-s3</artifactId>
    </dependency>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>dynamodb</artifactId>
    </dependency>
</dependencies>
```



# OpenPGP-Schlüssel für den AWS SDK for Java

Alle öffentlich verfügbaren Maven-Artefakte für AWS SDK for Java sind mit dem OpenPGP-Standard signiert. Der öffentliche Schlüssel, den Sie zur Überprüfung der Signatur eines Artefakts benötigen, ist im folgenden Abschnitt verfügbar.

## Aktueller Schlüssel

Die folgende Tabelle enthält OpenPGP-Schlüsselinformationen für die aktuellen Versionen des SDK for Java 1.x und SDK for Java 2.x.

Schlüssel-ID	0xAC107B386692DADD
Typ	RSA
Größe	4096/4096
Erstellt	2016-06-30
Läuft ab	2024-10-08
Benutzer-ID	AWSSDKs und Tools <@amazon.com> aws-dr-tools
Schlüssel-Fingerabdruck	FEB9 209F 2F2F 3F46 6484 1E55 AC10 7B38 6692 HINZUFÜGEN

Um den folgenden öffentlichen OpenPGP-Schlüssel für das SDK for Java in die Zwischenablage zu kopieren, wählen Sie das Symbol „Kopieren“ in der oberen rechten Ecke.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
xsFNBFd1gAUBEACqbmmFbxkJgz1lD7wrlskQA1LLuSAC4p8ny9u/D2zLR8Ynk3Yz
mzJuQ+Kfjne2t+xTDex6MPJlMYp0viSWsX2psgvdmeyUpW9ap01rThNYkc+W5fRc
buFehfbi9LSATZGJi8RG0sCCr5FsYVz0gEk85M2+PeM24cXhQIOZtQUjswX/pdk/
KduGtZASqNAYLKR0mRODzUuaokLPo24pfm9bnr1RnRtw5ktPAA5bM9ZZaGKriej
kT2lPffBbjp8F5AZvmGLtNm2Cmg4FKBvI04SQjy2jjrQ3wBzi5Lc9HTxDuHK/rtV
u6PewUe2WP1nxlXenhMZU1UK4YoSB9E9StQ2VxQiySLHSdxR7Ma4WgYdVLn9b0ie
```

```
nj3QxLuQ1ZUKF79ES6JaM4t0z1gGcQeU1+Uk1gjFLuKwmzWRdEIFfxMyvH6qgKnd
U+DioH5mcUwhwffAAsuIJyAdMIEUYh7IfzJJXQf+ff+Xf0Cl6by0JFWrIGQkAzMu
CEvaCfwtHC2Lpzo33/WRFEMAuzzd0QJ4uz4xFFvaS0SZHMLHWI9YV/+Pea3X99Ms
0Nlek/LolAJh67MynHeVB0HKrq+fluorWepQivctzN6Y1N0kx5naTPGGaKWK7G2q
TbcY5SMnkIwFLFSougj0Fvmjczq8iZRwYxWA+i+LQvsR9WEXEiQffIWRoQARAQAB
zsFNBFd1gAUBEAC8zNARpWb3dPMThL2xAY+fS60vXdB1Sk0tYJpDWPfGvo0d+VQ+
hV6Xu1GAHAS6xG1WHysPT9KejIRSgLG+e9CaM5yhsxNa1WFGUM4Q9ESo3t+a75Go
7xHIxgFjC046/06Vh3g9N/PREeuG8zkZ3H2v5fmD+ejyPgk4W9sFL00zjRiZD0FK
VYR/j9uenEC/2NBcLuFy3q6cDfmCoDE0062kXMnaGz3knzEK/X1SkcjsxRDq7zaQ
lQ1Kou+3dICwy4x5SJQ8j1+eeeEvF2C2/dXmDohb57tqUwioohMUQkmCtvZgEHjy
pUwgp0MTo25gWxkvJlSJKU0b6b1786WnySIzF2gxqlkkEmB14RAssQkeXjrSmGws
MDyHNqyJeYFus18sPaSPO+V2n0z+2B070Uq+wmf1S5A5FpegH0PZzzoNZo8I6Qxa
Zje9YSZUijGmZIdEBleRVt3Svhi8MY1nasd4bW2RK1sr7plkBf8QRe6biiQRf3KD
0Sn5CbmXpAcHJ1ZHzRRdkXZDNQC6vCJxsy1300TrhJtAV1Yq347uyUbVi291ISVg
roUVtprismHoEk5Go0THbg9SCSt+xi/FiJQC+ubWmIGXoFKMR3UmhDnnzobKcbtnbs
/Hd981FdVghYYvq//gTakJk0WxfGq030wtXRndPOA0T+qhP3TE+LtGRJ+wARAQAB
wsF1BBgBCgAPBQJXdYAFaHsMBQkHhh+AAAoJEKwQezhmktrdTyEP/0H0VWHwQsaW
jMrGj000MFzXGUo8SBmYYTBs29VM8wBGDsPkYCjeZzU16i9iqDpDqxyqmTigcjH
V8CDx/6xsMBLG2yKaKZ4m3+Yn0Qf/sQkyCvqiyMF9mS7pDYWy+mPhPuw8TDIfiqg
VhzjSpIMFWPqxVjn6KKbPN/QASr3Pf0cuP6qpHG+NAM6Q5dYkCebyvwzLmg1sVni
l6iSyJd1jBj3D34XrgWS9buyxBB2CjIM76WxfNViJ9zAaPI78X9v6PpDGn0kg6oL
zrusrvBjoZknKQm0SZ+41fx6xvrTPs8uPEzevzJB1kke6kw9+KagY8mrVX1ZenRg
+sY/4vxJreYWQeq167ggx+wFjKDcfhZA7m70LH0DysrGVCLcmuinUBaN1HmLDcGY
XZ+kMCoXf0bpuCVByQmNJgEb47EIFlx/+TEeNHKM0+22xL1atFzXfkEVZck+NghL
ZyFDhS3g1bma7puU7r752uiJjA6Iv8+kHDXi+/V7GNpuiEFUYh69QQ2//CS5H51o
sC/Bkb9evSn/Lp8dMubtWAaXDGJMgw9vqZ55N02NK0fvF/IKHnGkvH28rv00PCv0
WTA/MClv28y0PrSvvcMXnduLtkBEX7TISMPW+n+0Ta63/z4YFFEZ7sFLrEm3Q3vJ
MN3mE5i3cw+JGXPSu0nTtgqk/oZv//SS
=Z9u3
-----END PGP PUBLIC KEY BLOCK-----
```

# Dokumentverlauf

In diesem Thema werden wichtige Änderungen am - AWS SDK for Java Entwicklerhandbuch im Laufe seines Verlaufs beschrieben.

Dieses Handbuch wurde zuletzt am 4. März 2024 veröffentlicht.

Änderung	Beschreibung	Datum
<a href="#">the section called “Sicheres”</a>	Fügen Sie Anweisungen zum Deaktivieren von IMDSv1 hinzu.	14. März 2024
<a href="#">the section called “S-step-by-step Anweisungen”</a>	Fügen Sie step-by-step Migrationsanweisungen hinzu.	8. März 2024
<a href="#">Migrieren zu Version 2</a>	Aktualisierung des Migration sthemas.	14. Februar 2024
<a href="#">the section called “AWSCRT-basierte HTTP-Clients konfigurieren”</a>	Fügen Sie Informationen über den synchronen AWS CRT-basierten HTTP-Client hinzu.	5. Januar 2024
<a href="#">the section called “Amazon Cognito Identity”</a> und <a href="#">the section called “Amazon Cognito Identity Provider”</a>	Amazon Cognito-Beispiele wurden in den Abschnitt Codebeispiele verschoben.	28. Dezember 2023
<a href="#">SDK-Funktionen verwenden</a>	Das Thema SDK-Features wurde überarbeitet.	11. Dezember 2023
<a href="#">OpenPGP-Schlüssel</a>	Geben Sie den aktuellen OpenPGP-Schlüssel an.	6. Dezember 2023
<a href="#">the section called “Serialisierungsänderungen”</a>	Beschreiben Sie die Serialisierungsunterschiede zwischen v1 und v2 des SDK for Java.	05. Dezember 2023

Änderung	Beschreibung	Datum
<a href="#">the section called “S3 Transfer Manager”</a>	Fügen Sie einen Abschnitt hinzu, in dem die Änderungen im S3 Transfer Manager von Version 1 zu Version 2 detailliert beschrieben werden.	13. November 2023
<a href="#">the section called “Annotationsreferenz”</a>	Fügen Sie eine Liste von Datenklassen-Anmerkungen hinzu, die mit dem DynamoDB Enhanced Client verwendet werden können.	30. Oktober 2023
<a href="#">???</a>	Hinzufügen von Informationen zum Migrationsstatus von Bibliotheken und Dienstprogrammen von SDK for Java v1.x zu v2.x	17. Oktober 2023
<a href="#">???</a>	Aktualisieren des Gradle-Setup-Themas	17. Oktober 2023
<a href="#">the section called “Ignoriere Null-Attribute verschachtelter Objekte”</a>	Fügen Sie Informationen zur DynamoDB Enhanced Client-@DynamoDb IgnoreNulls Anmerkung hinzu.	22. September 2023
<a href="#">the section called “Regionsübergreifender Zugriff”</a>	Fügen Sie Informationen zum regionsübergreifenden Zugriff auf Amazon S3-Buckets hinzu.	31. August 2023
<a href="#">the section called “Leere Objekte beibehalten”</a>	Fügen Sie einen Abschnitt hinzu, der die @DynamoDb PreserveEmptyObject Anmerkung behandelt.	25. August 2023

Änderung	Beschreibung	Datum
<a href="#">???</a>	Abschnitt Service-Client aktualisieren.	15. August 2023
<a href="#">the section called “Empfehlungen von Kunden”</a>	Seit Version 0.23 unterstützt AWS CRT ein musl-basiertes Betriebssystem wie Alpine Linux. HTTP-Client-Empfehlungen spiegeln jetzt die Unterstützung von Musl wider.	11. August 2023
<a href="#">the section called “Erstellen von IAM-Richtlinien”</a>	Hinzufügen des API-Abschnitts von IAM Policy Builder	31. Juli 2023
<a href="#">the section called “Erste Schritte”</a>	Korrigieren Sie mehrere Ausschnitte im Abschnitt Erste Schritte des Themas DynamoDB Enhanced Client.	24. Juli 2023
<a href="#">the section called “Proxy-Unterstützung”</a>	Fügen Sie HTTP-Proxy-Supportinformationen und Beispiele für jeden HTTP-Client hinzu.	02. Juni 2023
Reorganisieren des Inhaltsverzeichnis	Abschnitt <a href="#">Codebeispiele</a> und <a href="#">Arbeiten mit AWS-Services</a> zu TOC-Einträgen der obersten Ebene hochstufen.	24. Mai 2023
<a href="#">the section called “Protokollierungsabhängigkeit hinzufügen”</a>	Abschnitt zum Anzeigen von Gradle-Abhängigkeiten im Protokollierungsbereich.	23. Mai 2023
<a href="#">the section called “Arbeiten mit paginierten Ergebnissen”</a>	Paginierungsthema aktualisieren.	18. Mai 2023

Änderung	Beschreibung	Datum
<a href="#">the section called “Richten Sie ein Gradle-Projekt ein”</a>	Aktualisieren Sie die Einrichtung des Gradle-Projekts.	3. Mai 2023
<a href="#">DynamoDB Enhanced Client API</a>	Das Thema DynamoDB Enhanced Client API wurde neu geschrieben.	28. April 2023
<a href="#">Aktualisieren der Anweisungen für das Tutorial „Erste Schritte“</a>	Maven archetype wurde geändert, um die Option für credentialsProvider einzuschließen; Anweisungen wurden entsprechend geändert.	11. April 2023
<a href="#">the section called “Empfehlungen von Kunden”</a>	Unterstützung für HTTP-Client-Entscheidungen hinzufügen	30. März 2023
Aktualisierungen der bewährten Methoden für IAM	Aktualisierter Leitfaden, angepasst an die bewährten IAM-Methoden. Weitere Informationen finden Sie unter <a href="#">Bewährte IAM-Methoden</a> .	14. März 2023
<a href="#">the section called “Profilanmeldeinformationen neu laden”</a>	Abschnitt zum Neuladen von Profilanmeldeinformationen hinzufügen.	9. Februar 2023
<a href="#">the section called “AWSCRT-basierte HTTP-Clients konfigurieren”</a>	Aktualisierungsthema für die GA-Version.	8. Februar 2023
<a href="#">the section called “Mit Amazon EC2 EC2-Instance-Metadaten arbeiten”</a>	Fügen Sie ein geführtes Beispiel für den Java-SDK-Client für den Amazon S3-Instance-Metadatenservice hinzu.	1. Februar 2023

Änderung	Beschreibung	Datum
<a href="#">the section called “Verwenden Sie einen performanten S3-Client”</a>	Abschnitt für den AWS CRT-basierten S3-Client hinzufügen.	19. Dezember 2022
<a href="#">the section called “Übertragen von Dateien und Verzeichnissen”</a>	Aktualisieren Sie Beispiele für Amazon S3 Transfer Manager für die GA-Version.	19. Dezember 2022
<a href="#">the section called “Bewährte Methoden”</a>	Abschnitt „Bewährte Methoden“ hinzugefügt.	18. November 2022
<a href="#">the section called “Laden temporärer Anmeldeinformationen aus einem externen Prozess”</a>	Abschnitt zum Laden von Anmeldeinformationen aus einem externen Prozess hinzugefügt.	15. November 2022
<a href="#">the section called “Service-Client-Metriken”</a>	Aktualisierte Metrikaufistung mit HTTP-Client-Nutzungsanforderung.	9. November 2022
<a href="#">the section called “Übertragen von Dateien und Verzeichnissen”</a>	Beispielcode korrigiert.	02. November 2022
<a href="#">the section called “Reduzieren Sie die Startzeit des SDK für AWS Lambda”</a>	Aktualisierter Abschnitt mit zusätzlichen Optionen zur Reduzierung der Lambda-Startzeit.	1. November 2022
<a href="#">the section called “HTTP-Clients”</a>	Es wurden Konfigurationsinformationen hinzugefügt, um alle HTTP-Clients im SDK abzudecken.	26. Oktober 2022

Änderung	Beschreibung	Datum
<a href="#">the section called “Protokollierung”</a>	Das Protokollierungsthema wurde aktualisiert, um Details zur Wire-Protokollierung für alle HTTP-Clients aufzunehmen.	4. Oktober 2022
<a href="#">the section called “AWS - Datenbankservices”</a>	Überblick über AWS Datenbankservices und das SDK for Java 2.x hinzugefügt.	13. September 2022
<a href="#">EC2-Classic-Netzwerke werden eingestellt</a>	EC2-Classic wird am 15. August 2022 eingestellt.	28. Juli 2022
<a href="#">the section called “Zusätzliche Authentifizierungsoptionen”</a>	Aktualisierung der Abhängigkeit, die für die Single-Sign-On-Authentifizierung erforderlich ist.	18. Juli 2022
<a href="#">the section called “Transport Layer Security (TLS)”</a>	Aktualisieren Sie die TLS-Sicherheitsinformationen.	8. April 2022
<a href="#">the section called “Zusätzliche Authentifizierungsoptionen”</a>	Weitere Informationen zum Einrichten und Verwenden von Anmeldeinformationen wurden hinzugefügt.	22. Februar 2021
<a href="#">the section called “Richten Sie ein GraalVM Native Image-Projekt ein”</a>	Neues Thema zum Einrichten eines GraalVM Native Image-Projekts.	18. Februar 2021
<a href="#">the section called “Abfrage nach Ressourcenstatus”</a>	Writers veröffentlicht; Thema für die neue Funktion hinzugefügt.	30. September 2020
<a href="#">the section called “SDK-Metriken verwenden”</a>	Veröffentlichte Metriken; Thema für das neue Feature hinzugefügt.	17. August 2020



Änderung	Beschreibung	Datum
<a href="#">the section called “Amazon SNS”</a>	Es wurden Beispielthemen für hinzugefügt Amazon SNS.	30. Mai 2020
<a href="#">the section called “Reduzieren Sie die Startzeit des SDK für AWS Lambda”</a>	Thema AWS Lambda zur Funktionsleistung hinzugefügt.	29. Mai 2020
<a href="#">the section called “Festlegen der JVM-TTL für DNS-Namensuche”</a>	Das JVM-TTL-DNS-Caching-Thema wurde hinzugefügt.	27. April 2020
<a href="#">the section called “Richten Sie ein Apache Maven-Projekt ein”, the section called “Richten Sie ein Gradle-Projekt ein”</a>	Neue Maven- und Gradle-Einrichtungsthemen.	21. April 2020
<a href="#">the section called “Transport Layer Security (TLS)”</a>	Hinzufügung von TLS 1.2 zum Sicherheitsabschnitt.	19. März 2020
<a href="#">the section called “Abonnieren von Amazon Kinesis Data Streams”</a>	Stream Kinesis -Beispiele hinzugefügt.	2. August 2018
<a href="#">the section called “Arbeiten mit paginierten Ergebnissen”</a>	Thema der automatischen Paginierung hinzugefügt.	5. April 2018
<a href="#">???</a>	Es wurden Beispielthemen für IAM, Amazon EC2 CloudWatch und hinzugefügt DynamoDB.	29. Dezember 2017
<a href="#">the section called “Amazon S3”</a>	Beispiel für getObject für hinzugefügt Amazon S3.	7. August 2017
<a href="#">the section called “Asynchrone Programmierung verwenden”</a>	Asynchron-Thema hinzugefügt.	4. August 2017

Änderung	Beschreibung	Datum
GA-Version von <a href="#">AWS SDK for Java 2.x</a>	AWS SDK for Java Version 2 (v2) veröffentlicht.	28. Juni 2017

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.