



Entwicklerhandbuch für Version 1.x

# AWS SDK für Java 1.x



# AWS SDK für Java 1.x: Entwicklerhandbuch für Version 1.x

# Table of Contents

.....	viii
AWS SDK für Java 1.x .....	1
Version 2 des SDK wurde veröffentlicht .....	1
Zusätzliche Dokumentation und Ressourcen .....	1
Unterstützung für Eclipse-IDE .....	2
Entwicklung von Anwendungen für Android .....	2
Anzeigen des SDK-Versionsverlaufs .....	2
Erstellen von Java-Referenzdokumentationen für frühere SDK-Versionen .....	2
Erste Schritte .....	4
Grundlegende Einrichtung .....	4
-Übersicht .....	4
Anmeldemöglichkeit beim AWS Zugriffsportal .....	5
Richten Sie gemeinsam genutzte Konfigurationsdateien ein .....	5
Installieren Sie eine Java-Entwicklungsumgebung .....	7
Möglichkeiten, das zu bekommen AWS SDK für Java .....	7
Voraussetzungen .....	7
Verwenden Sie ein Build-Tool .....	8
Laden Sie das vorgefertigte JAR herunter .....	8
Aus dem Quellcode erstellen .....	9
Verwenden Sie Build-Tools .....	10
Das SDK mit Apache Maven verwenden .....	10
Das SDK mit Gradle verwenden .....	13
Temporäre Zugangsdaten und Region .....	17
Konfigurieren Sie temporäre Anmeldeinformationen .....	18
Aktualisieren von IMDS-Anmeldeinformationen .....	19
Stellen Sie das ein AWS-Region .....	19
Verwendung der AWS SDK für Java .....	21
Bewährte Methoden für AWS Entwicklung mit dem AWS SDK für Java .....	21
S3 .....	21
Erstellen von Service-Clients .....	22
Abruf eines Client-Generators .....	23
Erstellen von Async-Clients .....	24
Verwenden DefaultClient .....	25
Client-Lebenszyklus .....	25

Geben Sie temporäre Anmeldeinformationen ein .....	25
Verwenden der standardmäßigen Anbieterkette von Anmeldeinformationen .....	26
Geben Sie einen Anbieter oder eine Anbieterkette für Anmeldeinformationen an .....	30
Geben Sie explizit temporäre Anmeldeinformationen an .....	30
Weitere Infos .....	31
AWS-Region Auswahl .....	31
Überprüfung der Serviceverfügbarkeit in einer Region .....	31
Auswählen einer Region .....	32
Auswahl eines bestimmten Endpunkts .....	32
Ermitteln Sie die Region automatisch anhand der Umgebung .....	33
Umgang mit Ausnahmen .....	34
Warum ungeprüfte Ausnahmen? .....	35
AmazonServiceException (und Unterklassen) .....	35
AmazonClientException .....	36
Asynchrone Programmierung .....	36
Java-Futures .....	36
Asynchrone Callbacks .....	38
Bewährte Methoden .....	40
Protokollierung AWS SDK für Java Calls .....	40
Herunterladen der Log4J-JAR .....	41
Festlegen des Klassenpfads .....	41
Service-Specific Fehler und Warnungen .....	42
Request/Response Protokollierung im Überblick .....	42
Verbose-Protokollierung des Netzwerkverkehrs .....	43
Protokollieren von Latenz-Metriken .....	44
Client-Konfiguration .....	45
Proxy-Konfiguration .....	45
HTTP-Transport-Konfiguration .....	45
TCP-Socketpuffer-Größenhinweise .....	47
Zugriffskontrollrichtlinien .....	48
Amazon S3 Beispiel .....	48
Amazon SQS Beispiel .....	49
Beispiel für Amazon SNS .....	49
Legen Sie die JVM-TTL für die Suche nach DNS-Namen fest .....	50
Wie legt man die JVM-TTL fest .....	50
Aktivierung von Metriken für AWS SDK für Java .....	52

Wie aktiviert man die Generierung von Java-SDK-Metriken .....	52
Verfügbare Arten von Metriken .....	53
Weitere Informationen .....	56
Codebeispiele .....	58
AWS SDK für Java 2.x .....	58
Amazon CloudWatch Beispiele .....	58
Metriken abrufen von CloudWatch .....	59
Veröffentlichen benutzerdefinierter Metrikdaten .....	61
Mit CloudWatch Alarmen arbeiten .....	62
Verwenden von Alarmaktionen in CloudWatch .....	65
Ereignisse senden an CloudWatch .....	67
Amazon DynamoDB Beispiele .....	70
Verwenden Sie AWS kontobasierte Endpunkte .....	70
Arbeiten mit Tabellen in DynamoDB .....	71
Arbeiten mit Elementen in DynamoDB .....	78
Amazon EC2 Beispiele .....	85
Tutorial: Eine EC2 Instanz starten .....	86
Verwenden von IAM-Rollen, um Zugriff auf AWS Ressourcen zu gewähren für Amazon EC2 .....	91
Tutorial: Amazon EC2 Spot-Instances .....	98
Tutorial: Erweitertes Amazon EC2 Spot-Anforderungsmanagement .....	110
Amazon EC2 Instanzen verwalten .....	127
Verwendung von Elastic IP-Adressen in Amazon EC2 .....	133
Regionen und Verfügbarkeitszonen verwenden .....	136
Mit Amazon EC2 Schlüsselpaaren arbeiten .....	139
Arbeiten mit Sicherheitsgruppen in Amazon EC2 .....	141
AWS Identity and Access Management (IAM) Beispiele .....	145
Verwalten von IAM-Zugriffsschlüsseln .....	145
Verwalten von IAM-Benutzern .....	150
Verwenden von IAM-Konto-Aliasen .....	153
Arbeiten mit IAM-Richtlinien .....	156
Arbeiten mit IAM-Serverzertifikaten .....	161
Lambda Amazon-Beispiele .....	164
Serviceoperationen .....	165
Amazon Pinpoint Beispiele .....	169
Apps erstellen und löschen in Amazon Pinpoint .....	169

Endpunkte erstellen in Amazon Pinpoint .....	171
Segmente erstellen in Amazon Pinpoint .....	173
Kampagnen erstellen in Amazon Pinpoint .....	175
Kanäle aktualisieren in Amazon Pinpoint .....	176
Amazon S3 Beispiele .....	178
Amazon S3 Buckets erstellen, auflisten und löschen .....	178
Operationen an Amazon S3 Objekten ausführen .....	184
Amazon S3 Zugriffsberechtigungen für Buckets und Objekte verwalten .....	189
Verwaltung des Zugriffs auf Amazon S3 Buckets mithilfe von Bucket-Richtlinien .....	193
TransferManager Für Amazon S3 Operationen verwenden .....	197
Einen Amazon S3 Bucket als Website konfigurieren .....	210
Amazon S3 Clientseitige Verschlüsselung verwenden .....	213
Amazon SQS Beispiele .....	220
Mit Amazon SQS Nachrichtenwarteschlangen arbeiten .....	220
Amazon SQS Nachrichten senden, empfangen und löschen .....	223
Long Polling für Amazon SQS Nachrichtenwarteschlangen aktivieren .....	226
Sichtbarkeits-Timeout einrichten in Amazon SQS .....	228
Verwenden von Warteschlangen für unzustellbare Briefe in Amazon SQS .....	231
Amazon SWF Beispiele .....	233
SWF-Grundlagen .....	234
Eine einfache Amazon SWF Anwendung erstellen .....	236
Lambda Aufgaben .....	256
Korrektes Herunterfahren von Aktivitäts- und Workflow-Workern .....	261
Registrieren von Domänen .....	264
Auflisten von Domänen .....	265
Im SDK enthaltene Codebeispiele .....	266
Abrufen der Beispiele .....	266
Erstellen und Ausführen der Beispiele in der Befehlszeile .....	266
Erstellen und Ausführen der Beispiele in der Eclipse-IDE .....	267
Sicherheit .....	269
Datenschutz .....	270
Erzwingen einer Mindest-TLS-Version .....	271
Vorgehensweise zum Überprüfen der TLS-Version .....	271
Erzwingen einer Mindest-TLS-Version .....	271
Identitäts- und Zugriffsverwaltung .....	272
Zielgruppe .....	272

---

Authentifizierung mit Identitäten .....	273
Verwalten des Zugriffs mit Richtlinien .....	274
Wie AWS-Services arbeiten Sie mit IAM .....	276
Fehlerbehebung AWS Identität und Zugriff .....	276
Compliance-Validierung .....	279
Ausfallsicherheit .....	279
Infrastruktursicherheit .....	280
Migration des S3-Verschlüsselungsclients .....	280
Voraussetzungen .....	281
Überblick über die Migration .....	281
Aktualisieren Sie bestehende Clients, sodass sie neue Formate lesen können .....	281
Migrieren Sie Verschlüsselungs- und Entschlüsselungsclients auf V2 .....	282
Weitere Beispiele .....	285
OpenPGP-Schlüssel .....	287
Aktueller Schlüssel .....	287
Frühere Schlüssel .....	293
Dokumentverlauf .....	300

Für Version AWS SDK für Java 1.x wurde der Support am 31. Dezember 2025 eingestellt. Wir empfehlen Ihnen, auf den zu migrieren, [AWS SDK for Java 2.x](#) um weiterhin neue Funktionen, Verfügbarkeitsverbesserungen und Sicherheitsupdates zu erhalten.

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.

# Entwicklerhandbuch — AWS SDK für Java 1.x

Das [AWS SDK für Java](#) bietet eine Java-API für AWS Dienste. Mit dem SDK können Sie auf einfache Weise Java-Anwendungen erstellen, die mit Amazon S3, Amazon EC2 DynamoDB, und mehr funktionieren. Unterstützung für neue Services ergänzen wir regelmäßig im AWS SDK für Java. Eine Liste der unterstützten Services und deren API-Versionen, die in den einzelnen Versionen des SDKs enthalten sind, finden Sie in den [Versionshinweisen](#) für die Version, mit der Sie arbeiten.

## Version 2 des SDK wurde veröffentlicht

Schauen Sie sich das neue AWS SDK für Java 2.x unter <https://github.com/aws/aws-sdk-java-v2/> an. Sie enthält mit Spannung erwartete Funktionen, wie z. B. die Möglichkeit, eine HTTP-Implementierung einzubinden. Informationen zu den ersten Schritten finden Sie im [AWS SDK für Java 2.x Developer Guide](#).

## Zusätzliche Dokumentation und Ressourcen

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

- [AWS SDK für Java API Reference](#)
- [Java-Entwicklerblog](#)
- [Java-Entwicklerforen](#)
- GitHub:
  - [Dokumentationsquelle](#)
  - [Dokumentationsprobleme](#)
  - [SDK-Quellcode](#)
  - [SDK-Probleme](#)
  - [SDK-Beispiele](#)
  - [Gitter-Kanal](#)
- Die [AWS-Codebeispiel-Katalog](#)
- [@awsforjava \(Twitter\)](#)
- [Versionshinweise](#)

## Unterstützung für Eclipse-IDE

Wenn Sie Code mit der Eclipse-IDE entwickeln, können Sie das verwenden, [AWS Toolkit for Eclipse](#) um das AWS SDK für Java zu einem vorhandenen Eclipse-Projekt hinzuzufügen oder ein neues AWS SDK für Java Projekt zu erstellen. Das Toolkit unterstützt auch das Erstellen und Hochladen von Lambda Funktionen, das Starten und Überwachen von Amazon EC2 Instanzen, das Verwalten von IAM Benutzern und Sicherheitsgruppen, einen AWS CloudFormation Vorlageneditor und mehr.

Die vollständige Dokumentation finden Sie im [AWS Toolkit for Eclipse Benutzerhandbuch](#).

## Entwicklung von Anwendungen für Android

Wenn Sie ein Android-Entwickler sind, Amazon Web Services veröffentlicht ein SDK, das speziell für die Android-Entwicklung entwickelt wurde: das [Amplify Android \(AWS Mobile SDK for Android\)](#).

## Anzeigen des SDK-Versionsverlaufs

Den Versionsverlauf von AWS SDK für Java, einschließlich der Änderungen und unterstützten Dienste pro SDK-Version, finden Sie in den [Versionshinweisen](#) des SDK.

## Erstellen von Java-Referenzdokumentationen für frühere SDK-Versionen

Die [AWS SDK für Java API-Referenz](#) stellt den neuesten Build der Version 1.x des SDK dar. Wenn Sie einen früheren Build der 1.x-Version verwenden, möchten Sie möglicherweise auf die SDK-Referenzdokumentation zugreifen, die der von Ihnen verwendeten Version entspricht.

Die einfachste Methode zum Erstellen der Dokumentation besteht darin, das Build-Tool von Apache [Maven](#) zu nutzen. Laden Sie Maven zuerst herunter und installieren Sie es, falls es auf Ihrem System noch nicht vorhanden ist, und erstellen Sie die Referenzdokumentation dann mit den folgenden Schritten:

1. Suchen Sie auf der [Releases-Seite](#) des SDK-Repositorys die SDK-Version, die Sie verwenden, und wählen Sie sie aus. GitHub
2. Wählen Sie entweder den Link `zip` (die meisten Plattformen, einschließlich Windows) oder `tar.gz` (Linux, macOS oder Unix), um das SDK auf Ihren Computer herunterzuladen.

3. Extrahieren Sie das Archiv in ein lokales Verzeichnis.
4. Navigieren Sie in der Befehlszeile zu dem Verzeichnis, in das Sie das Archiv entpackt haben. Geben Sie dann folgenden Befehl ein:

```
mvn javadoc:javadoc
```

5. Nachdem die Erstellung abgeschlossen ist, finden Sie die generierte HTML-Dokumentation im Verzeichnis `aws-java-sdk/target/site/apidocs/`.

# Erste Schritte

Dieser Abschnitt enthält Informationen zur Installation, Einrichtung und Verwendung des AWS SDK für Java.

Themen

- [Grundkonfiguration zum Arbeiten AWS-Services](#)
- [Möglichkeiten, das zu bekommen AWS SDK für Java](#)
- [Verwenden Sie Build-Tools](#)
- [AWS Temporäre Anmeldeinformationen und AWS-Region für die Entwicklung einrichten](#)

## Grundkonfiguration zum Arbeiten AWS-Services

### -Übersicht

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

- Sie müssen in der Lage sein, [sich bei dem AWS Zugangsportaal anzumelden](#), das im verfügbar ist AWS IAM Identity Center.
- Die [Berechtigungen der für das SDK konfigurierten IAM-Rolle](#) müssen den Zugriff auf die AWS-Services , die Ihre Anwendung benötigt, ermöglichen. Die mit der PowerUserAccess AWS verwalteten Richtlinie verbundenen Berechtigungen reichen für die meisten Entwicklungsanforderungen aus.
- Eine Entwicklungsumgebung mit den folgenden Elementen:
  - [Gemeinsam genutzte Konfigurationsdateien](#), die auf folgende Weise eingerichtet werden:
    - Die `config` Datei enthält ein Standardprofil, das eine spezifiziert AWS-Region.
    - Die `credentials` Datei enthält temporäre Anmeldeinformationen als Teil eines Standardprofils.
  - Eine geeignete [Installation von Java](#).
  - Ein [Tool zur Build-Automatisierung](#) wie [Maven](#) oder [Gradle](#).
  - Ein Texteditor für die Arbeit mit Code.
  - (Optional, aber empfohlen) Eine IDE (integrierte Entwicklungsumgebung) wie [IntelliJ IDEA](#), [Eclipse](#) oder. [NetBeans](#)

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

### Important

Bei den Anweisungen in diesem Abschnitt zur Einrichtung wird davon ausgegangen, dass Sie oder Ihr Unternehmen IAM Identity Center verwenden. Wenn Ihre Organisation einen externen Identitätsanbieter verwendet, der unabhängig von IAM Identity Center arbeitet, finden Sie heraus, wie Sie temporäre Anmeldeinformationen für das SDK for Java erhalten können. Folgen Sie [diesen 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 `[default]` so lautet, dass Sie dem SDK keinen Profilnamen angeben müssen oder AWS CLI.

## Anmeldemöglichkeit beim AWS Zugriffsportal

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

Wenn Sie mit dem AWS Zugriffsportal nicht vertraut sind, folgen Sie den Anweisungen für den Kontozugriff in [Schritt 1 des Themas IAM Identity Center-Authentifizierung](#) im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch. Folgen Sie nicht Schritt 2, da AWS SDK für Java 1.x die automatische Token-Aktualisierung und den automatischen Abruf temporärer Anmeldeinformationen für das SDK, das in Schritt 2 beschrieben wird, nicht unterstützt.

## Richten Sie gemeinsam genutzte Konfigurationsdateien ein

Die gemeinsam genutzten Konfigurationsdateien befinden sich auf Ihrer Entwicklungs-Workstation und enthalten grundlegende Einstellungen, die von all AWS SDKs und der AWS Command Line Interface (CLI) verwendet werden. Die gemeinsam genutzten Konfigurationsdateien können [eine Reihe von Einstellungen](#) enthalten, aber in diesen Anweisungen werden die grundlegenden Elemente festgelegt, die für die Arbeit mit dem SDK erforderlich sind.

## Richten Sie die gemeinsam genutzte **config** Datei ein

Das folgende Beispiel zeigt den Inhalt einer gemeinsam genutzten config Datei.

```
[default]
region=us-east-1
output=json
```

Verwenden Sie für Entwicklungszwecke die Datei, die dem Ort AWS-Region [am nächsten](#) liegt, an dem Sie Ihren Code ausführen möchten. Eine [Liste der Regionalcodes](#), die Sie in der config Datei verwenden können, finden Sie in der Allgemeine Amazon Web Services-Referenz Anleitung. Die json Einstellung für das Ausgabeformat ist einer von [mehreren möglichen Werten](#).

Folgen Sie den Anweisungen [in diesem Abschnitt](#), um die config Datei zu erstellen.

## Richten Sie temporäre Anmeldeinformationen für das SDK ein

Nachdem Sie über das Zugriffsportal AWS Zugriff auf eine AWS-Konto und IAM-Rolle erhalten haben, konfigurieren Sie Ihre Entwicklungsumgebung mit temporären Anmeldeinformationen für den Zugriff durch das SDK.

Schritte zum Einrichten einer lokalen **credentials** Datei mit temporären Anmeldeinformationen

1. [Erstellen Sie eine gemeinsam genutzte credentials Datei](#).
2. Fügen Sie den folgenden Platzhaltertext in die credentials Datei ein, bis Sie funktionierende temporäre Anmeldeinformationen einfügen.

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

3. Speichern Sie die Datei. Die Datei `~/.aws/credentials` sollte jetzt auf Ihrem lokalen Entwicklungssystem vorhanden sein. Diese Datei enthält das [\[Standard\] -Profil](#), das das SDK for Java verwendet, wenn kein bestimmtes benanntes Profil angegeben ist.
4. [Melden Sie sich beim AWS Access-Portal](#) an.
5. Folgen Sie diesen Anweisungen unter der Überschrift [Manuelle Aktualisierung der Anmeldeinformationen](#), um die Anmeldeinformationen für die IAM-Rolle aus dem AWS Zugriffsportal zu kopieren.



- Sie müssen in der Lage sein, [sich bei dem AWS Zugangsportal anzumelden](#), das im verfügbar ist AWS IAM Identity Center.
- Eine geeignete [Installation von Java](#).
- Temporäre Anmeldeinformationen, die in Ihrer lokalen gemeinsam genutzten `credentials` Datei eingerichtet sind.

Anweisungen zur Einrichtung für die Verwendung des SDK for Java finden Sie im [the section called "Grundlegende Einrichtung"](#) Thema.

## Verwenden Sie ein Build-Tool, um Abhängigkeiten für das SDK for Java zu verwalten (empfohlen)

Wir empfehlen, Apache Maven oder Gradle mit Ihrem Projekt zu verwenden, um auf die erforderlichen Abhängigkeiten des SDK for Java zuzugreifen. [In diesem Abschnitt](#) wird beschrieben, wie Sie diese Tools verwenden.

## Laden Sie das SDK herunter und extrahieren Sie es (nicht empfohlen)

Wir empfehlen, dass Sie ein Build-Tool verwenden, um auf das SDK für Ihr Projekt zuzugreifen. Sie können jedoch ein vorgefertigtes JAR der neuesten Version des SDK herunterladen.

### Note

Weitere Informationen zum Herunterladen und Erstellen von früheren Versionen des SDKs finden Sie unter [Installieren von früheren Versionen des SDKs](#).

1. Laden Sie das SDK aus der ZIP-Datei herunter <https://sdk-for-java.amazonwebservices.com/latest/aws-java-sdk>.
2. Extrahieren Sie den Inhalt nach dem Herunterladen des SDKs in ein lokales Verzeichnis.

Das SDK enthält folgende Verzeichnisse:

- `documentation`- enthält die API-Dokumentation (auch im Internet verfügbar: [AWS SDK für Java API-Referenz](#)).
- `lib`- enthält die `.jar` SDK-Dateien.

- `samples-` enthält einen funktionierenden Beispielcode, der demonstriert, wie das SDK verwendet wird.
- `third-party/lib-` enthält Bibliotheken von Drittanbietern, die vom SDK verwendet werden, wie Apache Commons Logging, AspectJ und das Spring-Framework.

Um das SDK zu verwenden, fügen Sie den vollständigen Pfad zu den Verzeichnissen `lib` und `third-party` zu den Abhängigkeiten in Ihrer Build-Datei hinzu und fügen Sie sie zu Ihrem Java-CLASSPATH hinzu, um Ihren Code auszuführen.

## Frühere Versionen des SDK aus dem Quellcode erstellen (nicht empfohlen)

Nur die neueste Version des vollständigen SDK wird in vorgefertigter Form als herunterladbares JAR bereitgestellt. Sie können jedoch eine vorherige Version des SDKs mit Apache Maven (Open Source) erstellen. Maven lädt alle erforderlichen Abhängigkeiten, erstellt und installiert das SDK in einem Schritt. Besuchen Sie <http://maven.apache.org/>, um Installationsanweisungen und weitere Informationen zu erhalten.

1. Gehen Sie zur GitHub SDK-Seite unter: [AWS SDK für Java \(GitHub\)](#).
2. Wählen Sie das Tag aus, das der gewünschten SDK-Versionsnummer entspricht. Beispiel, `1.6.10`.
3. Klicken Sie auf die Schaltfläche `Download ZIP`, um die ausgewählte Version des SDKs herunterzuladen.
4. Extrahieren Sie die Datei in ein Verzeichnis auf Ihrem Entwicklungssystem. Bei vielen Systemen können Sie dazu den grafischen Datei-Manager oder das `unzip`-Dienstprogramm in einem Terminal-Fenster nutzen.
5. Navigieren Sie in einem Terminal-Fenster in das Verzeichnis, in das Sie die SDK-Quelldateien entpackt haben.
6. Erstellen und installieren Sie das SDK mit dem folgenden Befehl ([Maven](#) erforderlich):

```
mvn clean install -Dpgp.skip=true
```

Die resultierende `.jar`-Datei wird im `target`-Verzeichnis erstellt.

7. (Optional) Erstellen Sie die API-Referenz-Dokumentation mit dem folgenden Befehl:

```
mvn javadoc:javadoc
```

Die Dokumentation wird im Verzeichnis `target/site/apidocs/` erstellt.

## Verwenden Sie Build-Tools

Die Verwendung von Build-Tools hilft bei der Verwaltung der Entwicklung von Java-Projekten. Es sind mehrere Build-Tools verfügbar, aber wir zeigen, wie Sie mit zwei beliebten Build-Tools, Maven und Gradle, loslegen können. In diesem Thema erfahren Sie, wie Sie mit diesen Build-Tools die SDK for Java Java-Abhängigkeiten verwalten, die Sie für Ihre Projekte benötigen.

Themen

- [Das SDK mit Apache Maven verwenden](#)
- [Das SDK mit Gradle verwenden](#)

## Das SDK mit Apache Maven verwenden

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

### Note

Um die Anleitungen in diesem Thema nachzuvollziehen, sollten Sie Maven installiert haben. Wenn Maven noch nicht installiert ist, besuchen Sie <http://maven.apache.org/>, um es herunterzuladen und zu installieren.

## Erstellen eines neuen Maven-Pakets

Sie können ein einfaches Maven-Paket erstellen, indem Sie ein Terminal-Fenster (eine Befehlszeile) öffnen und Folgendes ausführen:

```
mvn -B archetype:generate \  
-DarchetypeGroupId=org.apache.maven.archetypes \  
-DgroupId=org.example.basicapp \  
-DartifactId=myapp
```

Ersetzen Sie `org.example.basicapp` mit dem vollen Paket-Namespace Ihrer Anwendung und `myapp` mit dem Projektnamen (wird für den Verzeichnisnamen Ihres Projekts übernommen).

Erstellt standardmäßig eine Projektvorlage für Sie unter Verwendung des [Schnellstart-Archetyps](#), der für viele Projekte ein guter Ausgangspunkt ist. Es sind noch mehr Archetypen verfügbar. Auf der [Maven-Archetypen-Seite findest du eine Liste der Archetypen](#), die im Paket enthalten sind. Sie können einen bestimmten Archetyp zur Nutzung auswählen, indem Sie das Argument `-DarchetypeArtifactId` an den Befehl `archetype:generate` anhängen. Zum Beispiel:

```
mvn archetype:generate \  
-DarchetypeGroupId=org.apache.maven.archetypes \  
-DarchetypeArtifactId=maven-archetype-webapp \  
-DgroupId=org.example.webapp \  
-DartifactId=mywebapp
```

### Note

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

## Konfigurieren des SDKs als Maven-Abhängigkeit

Um das AWS SDK für Java in Ihrem Projekt zu verwenden, müssen Sie es als Abhängigkeit in der `pom.xml` Datei Ihres Projekts deklarieren. Ab Version 1.9.0 können Sie [einzelne Komponenten](#) oder das [gesamte SDK](#) importieren.

### Angeben einzelner SDK-Module

Verwenden Sie zur Auswahl einzelner SDK-Module die AWS SDK für Java Stückliste (BOM) für Maven. Dadurch wird sichergestellt, dass die von Ihnen angegebenen Module dieselbe Version des SDK verwenden und dass sie miteinander kompatibel sind.

Um die BOM zu verwenden, fügen Sie der Datei `pom.xml` Ihrer Anwendung einen Abschnitt `<dependencyManagement>` hinzu. Fügen Sie dabei `aws-java-sdk-bom` als Abhängigkeit hinzu und geben Sie die SDK-Version an, die Sie nutzen möchten:

```
<dependencyManagement>  
  <dependencies>  
    <dependency>  
      <groupId>com.amazonaws</groupId>  
      <artifactId>aws-java-sdk-bom</artifactId>
```

```
<version>1.11.1000</version>
<type>pom</type>
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
```

Die neueste Version der AWS SDK für Java Stückliste, die auf Maven Central verfügbar ist, finden Sie unter: [com.amazonaws/ https://mvnrepository.com/artifact/ aws-java-sdk-bom](https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk-bom) Auf dieser Seite können Sie auch sehen, welche von der BOM verwalteten Module (Abhängigkeiten) Sie im Abschnitt `<dependencies>` der Datei `pom.xml` Ihres Projekts einfügen können.

Sie können jetzt einzelne Module aus dem SDK zur Nutzung in Ihrer Anwendung auswählen. Da Sie die SDK-Version bereits in der BOM deklariert haben, müssen Sie die Versionsnummer nicht mehr für jede Komponente angeben.

```
<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-s3</artifactId>
  </dependency>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-dynamodb</artifactId>
  </dependency>
</dependencies>
```

Sie können auch in der nachlesen, welche Abhängigkeiten Sie [AWS-Codebeispiel-Katalog](#) für eine bestimmte Aufgabe verwenden sollten. AWS-Service Weitere Informationen finden Sie in der POM-Datei unter einem bestimmten Servicebeispiel. Wenn Sie beispielsweise an den Abhängigkeiten für den AWS S3-Dienst interessiert sind, finden Sie das [vollständige Beispiel](#) unter GitHub. (Schau dir den Pom unter `/java/example_code/s3` an).

### Importieren aller SDK-Module

Wenn Sie das gesamte SDK als Abhängigkeit aufnehmen möchten, verwenden Sie nicht die BOM-Methode. Deklarieren Sie es stattdessen einfach wie folgt in `pom.xml`:

```
<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
```

```
<artifactId>aws-java-sdk</artifactId>
<version>1.11.1000</version>
</dependency>
</dependencies>
```

## Erstellen Ihres Projekts

Sobald Ihr Projekt fertig eingerichtet ist, können Sie es mit dem Maven-Befehl `package` erstellen:

```
mvn package
```

Dadurch wird die `0.jar`-Datei im Verzeichnis `target` angelegt.

## Erstellen des SDKs mit Maven

Sie können Apache Maven verwenden, um das SDK aus den Quellen zu erstellen. Laden Sie dazu [den SDK-Code von herunter GitHub](#), entpacken Sie ihn lokal und führen Sie dann den folgenden Maven-Befehl aus:

```
mvn clean install
```

## Das SDK mit Gradle verwenden

Um die SDK-Abhängigkeiten für Ihr [Gradle-Projekt](#) zu verwalten, importieren Sie die Maven-Stückliste für AWS SDK für Java in die Datei der Anwendung. `build.gradle`

### Note

Ersetzen Sie in den folgenden Beispielen **1.12.529** in der Build-Datei durch eine gültige Version von. AWS SDK für Java Suchen Sie die neueste Version im [zentralen Maven-Repository](#).

## Projekteinrichtung für Gradle 4.6 oder höher

[Seit Gradle 4.6](#) können Sie die verbesserte POM-Unterstützungsfunktion von Gradle verwenden, um Stücklistendateien (BOM) zu importieren, indem Sie eine Abhängigkeit von einer Stückliste deklarieren.

1. Wenn Sie Gradle 5.0 oder höher verwenden, fahren Sie mit Schritt 2 fort. Andernfalls aktivieren Sie die Funktion `IMPROVED_POM_SUPPORT` in der `settings.gradle`-Datei.

```
enableFeaturePreview('IMPROVED_POM_SUPPORT')
```

2. Fügen Sie die Stückliste zum Abschnitt Abhängigkeiten der Anwendungsdatei hinzu.  
`build.gradle`

```
...
dependencies {
    implementation platform('com.amazonaws:aws-java-sdk-bom:1.12.529')

    // Declare individual SDK dependencies without version
    ...
}
```

3. Geben Sie im Abschnitt `dependencies` (Abhängigkeiten) die SDK-Module an, die verwendet werden sollen. Im Folgenden ist beispielsweise eine Abhängigkeit für Amazon Simple Storage Service (Amazon S3) enthalten.

```
...
dependencies {
    implementation platform('com.amazonaws:aws-java-sdk-bom:1.12.529')
    implementation 'com.amazonaws:aws-java-sdk-s3'
    ...
}
```

Gradle löst mit den Informationen aus der BOM automatisch die richtige Version der SDK-Abhängigkeiten auf.

Das Folgende ist ein Beispiel für eine vollständige `build.gradle` Datei, die eine Abhängigkeit für Amazon S3 enthält.

```
group 'aws.test'
version '1.0-SNAPSHOT'

apply plugin: 'java'

sourceCompatibility = 1.8

repositories {
```

```
mavenCentral()
}

dependencies {
    implementation platform('com.amazonaws:aws-java-sdk-bom:1.12.529')
    implementation 'com.amazonaws:aws-java-sdk-s3'
}
```

### Note

Ersetzen Sie im vorherigen Beispiel die Abhängigkeit von Amazon S3 durch die Abhängigkeiten der AWS Dienste, die Sie in Ihrem Projekt verwenden werden. Die Module (Abhängigkeiten), die von der AWS SDK für Java BOM verwaltet werden, sind im [zentralen Maven-Repository](#) aufgeführt.

## Projekteinrichtung für Gradle-Versionen vor 4.6

Gradle-Versionen vor 4.6 verfügen über keine native Stücklistenunterstützung. Um AWS SDK für Java Abhängigkeiten für Ihr Projekt zu verwalten, verwenden Sie das [Abhängigkeitsverwaltungs-Plugin](#) von Spring für Gradle, um die Maven-Stückliste für das SDK zu importieren.

1. Fügen Sie das Plugin für die Abhängigkeitsverwaltung zur Datei Ihrer Anwendung hinzu.  
`build.gradle`

```
buildscript {
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath "io.spring.gradle:dependency-management-plugin:1.0.9.RELEASE"
    }
}

apply plugin: "io.spring.dependency-management"
```

2. Fügen Sie die BOM in den Abschnitt `dependencyManagement` der Datei ein.

```
dependencyManagement {
    imports {
        mavenBom 'com.amazonaws:aws-java-sdk-bom:1.12.529'
```

```
}  
}
```

3. Geben Sie im Abschnitt Abhängigkeiten die SDK-Module an, die Sie verwenden werden. Im folgenden Beispiel ist eine Abhängigkeit für Amazon S3 enthalten.

```
dependencies {  
    compile 'com.amazonaws:aws-java-sdk-s3'  
}
```

Gradle löst mit den Informationen aus der BOM automatisch die richtige Version der SDK-Abhängigkeiten auf.

Das Folgende ist ein Beispiel für eine vollständige `build.gradle` Datei, die eine Abhängigkeit für enthält Amazon S3.

```
group 'aws.test'  
version '1.0'  
  
apply plugin: 'java'  
  
sourceCompatibility = 1.8  
  
repositories {  
    mavenCentral()  
}  
  
buildscript {  
    repositories {  
        mavenCentral()  
    }  
    dependencies {  
        classpath "io.spring.gradle:dependency-management-plugin:1.0.9.RELEASE"  
    }  
}  
  
apply plugin: "io.spring.dependency-management"  
  
dependencyManagement {  
    imports {  
        mavenBom 'com.amazonaws:aws-java-sdk-bom:1.12.529'  
    }  
}
```

```
}  
  
dependencies {  
    compile 'com.amazonaws:aws-java-sdk-s3'  
    testCompile group: 'junit', name: 'junit', version: '4.11'  
}
```

### Note

Ersetzen Sie im vorherigen Beispiel die Abhängigkeit von Amazon S3 durch die Abhängigkeiten des AWS Dienstes, den Sie in Ihrem Projekt verwenden werden. Die Module (Abhängigkeiten), die von der AWS SDK für Java BOM verwaltet werden, sind im [zentralen Maven-Repository](#) aufgeführt.

Weitere Informationen über das Angeben von SDK-Abhängigkeiten mit der BOM finden Sie unter [Verwenden des SDK mit Apache Maven](#).

## AWS Temporäre Anmeldeinformationen und AWS-Region für die Entwicklung einrichten

Um mit dem eine Verbindung zu einem der unterstützten Dienste herzustellen AWS SDK für Java, müssen Sie AWS temporäre Anmeldeinformationen angeben. Die AWS SDKs und CLIs verwenden Anbieterketten, um an verschiedenen Stellen nach AWS temporären Anmeldeinformationen zu suchen, einschließlich system/user Umgebungsvariablen und lokalen AWS Konfigurationsdateien.

Dieses Thema enthält grundlegende Informationen zum Einrichten Ihrer AWS temporären Anmeldeinformationen für die lokale Anwendungsentwicklung mithilfe von AWS SDK für Java. Wenn Sie Anmeldeinformationen zur Nutzung innerhalb einer EC2-Instance einrichten möchten oder die Eclipse-IDE zur Entwicklung verwenden, sehen Sie sich stattdessen folgende Themen an:

- Wenn Sie eine EC2-Instance verwenden, erstellen Sie eine IAM-Rolle und gewähren Sie dann Ihrer EC2-Instance Zugriff auf diese Rolle, wie unter [Using IAM Roles to Grant Access to Resources on beschrieben](#). AWS Amazon EC2
- Richten Sie AWS Anmeldeinformationen in Eclipse mit dem ein. [AWS Toolkit for Eclipse](#) Weitere Informationen finden Sie im [AWS Toolkit for Eclipse Benutzerhandbuch](#) unter [AWS Anmeldeinformationen einrichten](#).

## Konfigurieren Sie temporäre Anmeldeinformationen

Sie können temporäre Anmeldeinformationen für AWS SDK für Java auf verschiedene Arten konfigurieren, aber hier sind die empfohlenen Vorgehensweisen:

- Legen Sie temporäre Anmeldeinformationen in der Profildatei für AWS Anmeldeinformationen auf Ihrem lokalen System fest, die sich unter folgender Adresse befindet:
  - `~/.aws/credentials` (Linux, MacOS und Unix)
  - `C:\Users\USERNAME\.aws\credentials` (Windows)

Anweisungen zum [the section called “Richten Sie temporäre Anmeldeinformationen für das SDK ein”](#) Abrufen Ihrer temporären Anmeldeinformationen finden Sie in diesem Handbuch.

- Legen Sie die `AWS_SESSION_TOKEN` Umgebungsvariablen `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, und fest.

Um diese Variablen auf Linux, macOS oder Unix festzulegen, verwenden Sie :

```
export AWS_ACCESS_KEY_ID=your_access_key_id
export AWS_SECRET_ACCESS_KEY=your_secret_access_key
export AWS_SESSION_TOKEN=your_session_token
```

In Windows können Sie die Variablen mit festlegen:

```
set AWS_ACCESS_KEY_ID=your_access_key_id
set AWS_SECRET_ACCESS_KEY=your_secret_access_key
set AWS_SESSION_TOKEN=your_session_token
```

- Für eine EC2-Instance: Geben Sie eine IAM-Rolle an und gewähren Sie Ihrer EC2-Instance Zugriff auf diese Rolle. Eine ausführliche Beschreibung der Funktionsweise finden Sie unter [IAM-Rollen für Amazon EC2](#) im Amazon EC2 Benutzerhandbuch für Linux-Instances.

Sobald Sie Ihre AWS temporären Anmeldeinformationen mit einer dieser Methoden eingerichtet haben, werden sie automatisch AWS SDK für Java mithilfe der standardmäßigen Anmeldeinformationsanbieterkette geladen. Weitere Informationen zum Arbeiten mit AWS Anmeldeinformationen in Ihren Java-Anwendungen finden Sie unter [Mit AWS Anmeldeinformationen arbeiten](#).

## Aktualisieren von IMDS-Anmeldeinformationen

Das AWS SDK für Java unterstützt die optionale Aktualisierung von IMDS-Anmeldeinformationen im Hintergrund alle 1 Minute, unabhängig von der Ablaufzeit der Anmeldeinformationen. Auf diese Weise können Sie Ihre Anmeldeinformationen häufiger aktualisieren und die Wahrscheinlichkeit verringern, dass die tatsächliche Verfügbarkeit beeinträchtigt wird, wenn Sie IMDS nicht erreichen. AWS

```
1. // Refresh credentials using a background thread, automatically every minute. This
   // will log an error if IMDS is down during
2. // a refresh, but your service calls will continue using the cached credentials
   // until the credentials are refreshed
3. // again one minute later.
4.
5. InstanceProfileCredentialsProvider credentials =
6.     InstanceProfileCredentialsProvider.createAsyncRefreshingProvider(true);
7.
8. AmazonS3Client.builder()
9.     .withCredentials(credentials)
10.    .build();
11.
12. // This is new: When you are done with the credentials provider, you must close it
   // to release the background thread.
13. credentials.close();
```

## Stellen Sie das ein AWS-Region

Sie sollten einen Standard festlegen AWS-Region , der für den Zugriff auf AWS Dienste mit dem verwendet wird AWS SDK für Java. Um die beste Netzwerkleistung zu erzielen, wählen Sie die Region aus, die geografisch in Ihrer Nähe (oder in der Nähe Ihrer Kunden) liegt. Eine Liste der Regionen für jeden Dienst finden Sie unter [Regionen und Endpunkte](#) in der Amazon Web Services allgemeinen Referenz.

### Note

Wenn Sie keine Region auswählen, wird standardmäßig us-east-1 verwendet.

Sie können ähnliche Techniken wie das Festlegen von Anmeldeinformationen verwenden, um Ihre AWS Standardregion festzulegen:

- Stellen Sie das AWS-Region in der AWS Konfigurationsdatei auf Ihrem lokalen System ein, die sich unter folgendem Pfad befindet:
  - ~/.aws/config unter Linux, macOS oder Unix
  - C:\Users\USERNAME\.aws\config unter Windows

Diese Datei sollte Zeilen im folgenden Format enthalten:

+

```
[default]
region = your_aws_region
```

+

Ersetzen Sie AWS-Region `your_aws_region` durch Ihre gewünschte (z. B. „us-east-1“).

- Legen Sie die `AWS_REGION`-Umgebungsvariable fest.

Verwenden Sie unter Linux, macOS oder Unix :

```
export AWS_REGION=your_aws_region
```

In Windows nutzen Sie :

```
set AWS_REGION=your_aws_region
```

Wobei `your_aws_region` der gewünschte Name ist. AWS-Region

# Verwendung der AWS SDK für Java

Dieser Abschnitt enthält wichtige allgemeine Informationen zur Programmierung mit dem AWS SDK für Java, die für alle Dienste gelten, die Sie möglicherweise mit dem SDK verwenden.

Informationen und Beispiele zur dienstspezifischen Programmierung (für Amazon EC2, Amazon S3, Amazon SWF, usw.) finden Sie unter [AWS SDK für Java Codebeispiele](#).

## Themen

- [Bewährte Methoden für AWS Entwicklung mit dem AWS SDK für Java](#)
- [Erstellen von Service-Clients](#)
- [Geben Sie temporäre Anmeldeinformationen für AWS SDK für Java](#)
- [AWS-Region Auswahl](#)
- [Umgang mit Ausnahmen](#)
- [Asynchrone Programmierung](#)
- [Protokollierung AWS SDK für Java Calls](#)
- [Client-Konfiguration](#)
- [Zugriffskontrollrichtlinien](#)
- [Legen Sie die JVM-TTL für die Suche nach DNS-Namen fest](#)
- [Aktivierung von Metriken für AWS SDK für Java](#)

## Bewährte Methoden für AWS Entwicklung mit dem AWS SDK für Java

Die folgenden bewährten Methoden können Ihnen helfen, Probleme oder Probleme bei der Entwicklung von AWS Anwendungen mit dem zu vermeiden AWS SDK für Java. Wir haben diese bewährten Methoden nach Service angeordnet.

### S3

#### Vermeiden ResetExceptions

Wenn Sie Objekte mithilfe Amazon S3 von Streams hochladen (entweder über einen AmazonS3 Client oder TransferManager), können Netzwerkverbindungs- oder Timeoutprobleme auftreten.

Standardmäßig schlagen Übertragungen bei AWS SDK für Java Wiederholungsversuchen fehl, indem der Eingabestream vor dem Start einer Übertragung markiert und dann vor einem erneuten Versuch zurückgesetzt wird.

Wenn der Stream das Markieren und Zurücksetzen nicht unterstützt, gibt das SDK eine Meldung aus, [ResetException](#) wenn vorübergehende Fehler auftreten und Wiederholungsversuche aktiviert sind.

### Bewährte Methode

Wir empfehlen, dass Sie Streams einsetzen, die Markieren und Zurücksetzen unterstützen.

Der zuverlässigste Weg, dies zu vermeiden, [ResetException](#) besteht darin, Daten mithilfe einer [Datei](#) oder bereitzustellen, mit der sie umgehen AWS SDK für Java können [FileInputStream](#), ohne durch Markierungs- und Reset-Beschränkungen eingeschränkt zu sein.

Wenn es sich bei dem Stream nicht um einen Stream handelt, [FileInputStream](#) aber das Markieren und Zurücksetzen unterstützt, können Sie das Markierungslimit mithilfe der `setReadLimit` Methode von [RequestClientOptions](#) festlegen. Der Standardwert beträgt 128 KB. Wenn Sie den Wert für das Leselimit auf ein Byte setzen, das über der Größe des Streams liegt, wird a zuverlässig vermieden [ResetException](#).

Beträgt die maximal erwartete Größe eines Streams beispielsweise 100 000 Bytes, legen Sie die Lesegrenze auf 100 001 (100 000 + 1) Bytes fest. Das Markieren und Zurücksetzen funktioniert immer für 100 000 oder weniger Bytes. Hinweis: Dies könnte bei einigen Streams dazu führen, dass die angegebene Anzahl an Bytes in den Arbeitsspeicher gepuffert wird.

## Erstellen von Service-Clients

Um Anfragen zu stellen Amazon Web Services, erstellen Sie zunächst ein Service-Client-Objekt. Die empfohlene Methode besteht darin, den Service-Client-Generator zu nutzen.

Jedes AWS-Service hat eine Serviceschnittstelle mit Methoden für jede Aktion in der Service-API. Beispielsweise wird die Dienstschnittstelle für DynamoDB benannt. [AmazonDynamoDBClient](#) Jede Service-Schnittstelle verfügt über einen entsprechenden Client-Generator, mit dem Sie eine Implementierung der Service-Schnittstelle erstellen können. Die Client-Builder-Klasse für DynamoDB ist benannt. [AmazonDynamoDBClientBuilder](#)

## Abruf eines Client-Generators

Um eine Instance des Client-Generators abzurufen, verwenden Sie die statische Factory-Methode `standard`, wie im folgenden Beispiel gezeigt.

```
AmazonDynamoDBClientBuilder builder = AmazonDynamoDBClientBuilder.standard();
```

Sobald Sie einen Generator haben, können Sie die Eigenschaften des Clients anpassen, indem Sie die vielen praktischen Setter in der Generator-API nutzen. Beispielsweise können Sie wie folgt eine benutzerdefinierte Region und einen benutzerdefinierten Anmeldeinformationsanbieter festlegen.

```
AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .withCredentials(new ProfileCredentialsProvider("myProfile"))
    .build();
```

### Note

Die praktischen `withXXX`-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. Sobald ein Client erstellt wurde, ist er unveränderlich und alle Aufrufe an `setRegion` oder `setEndpoint` schlagen fehl.

Ein Generator kann mehrere Clients mit der gleichen Konfiguration erstellen. Wenn Sie Ihre Anwendung entwerfen, sollten Sie daran denken, dass der Generator veränderlich und nicht threadsicher ist.

Der folgende Code verwendet den Generator als Factory für Client-Instances.

```
public class DynamoDBClientFactory {
    private final AmazonDynamoDBClientBuilder builder =
        AmazonDynamoDBClientBuilder.standard()
            .withRegion(Regions.US_WEST_2)
            .withCredentials(new ProfileCredentialsProvider("myProfile"));

    public AmazonDynamoDB createClient() {
        return builder.build();
    }
}
```

```
}
```

## [Der Builder stellt außerdem fließende Setter für ClientConfiguration und RequestMetricCollector eine benutzerdefinierte Liste von 2 bereit. RequestHandler](#)

Im Folgenden finden Sie ein vollständiges Beispiel, in dem sämtliche konfigurierbaren Eigenschaften überschrieben werden.

```
AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .withCredentials(new ProfileCredentialsProvider("myProfile"))
    .withClientConfiguration(new ClientConfiguration().withRequestTimeout(5000))
    .withMetricsCollector(new MyCustomMetricsCollector())
    .withRequestHandlers(new MyCustomRequestHandler(), new
MyOtherCustomRequestHandler)
    .build();
```

## Erstellen von Async-Clients

Der AWS SDK für Java hat asynchrone (oder asynchrone) Clients für jeden Dienst (außer Amazon S3) und einen entsprechenden asynchronen Client-Builder für jeden Dienst.

So erstellen Sie einen asynchronen DynamoDB-Client mit dem Standard `ExecutorService`

```
AmazonDynamoDBAsync ddbAsync = AmazonDynamoDBAsyncClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .withCredentials(new ProfileCredentialsProvider("myProfile"))
    .build();
```

Zusätzlich zu den Konfigurationsoptionen, die der synchrone (oder synchrone) Client-Builder unterstützt, ermöglicht Ihnen der asynchrone Client, benutzerdefinierte Optionen festzulegen, um die [ExecutorFactory](#) zu ändern, `ExecutorService` die der asynchrone Client verwendet. `ExecutorFactory` ist eine funktionale Schnittstelle, sodass sie mit Lambda-Ausdrücken und Methodenreferenzen in Java 8 zusammenarbeitet.

So erstellen Sie einen asynchronen Client mit einem benutzerdefinierten `Executor`

```
AmazonDynamoDBAsync ddbAsync = AmazonDynamoDBAsyncClientBuilder.standard()
    .withExecutorFactory(() -> Executors.newFixedThreadPool(10))
```

```
.build();
```

## Verwenden DefaultClient

Sowohl der synchrone als auch der asynchrone Client-Generator haben eine weitere Factory-Methode mit dem Namen `defaultClient`. Mit dieser Methode wird ein Service-Client mit der Standardkonfiguration erstellt, wobei die Standardanbieterkette zum Laden von Anmeldeinformationen und der verwendet wird AWS-Region. Wenn die Anmeldeinformationen oder die Region nicht aus der Umgebung, in der die Anwendung ausgeführt wird, ermittelt werden können, schlägt der Aufruf von `defaultClient` fehl. Weitere Informationen darüber, wie [AWS Anmeldeinformationen und Region bestimmt werden, finden Sie unter Mit Anmeldeinformationen arbeiten](#) und [AWS-Region Auswahl](#).

### So erstellen Sie einen Standard-Service-Client

```
AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();
```

## Client-Lebenszyklus

Service-Clients im SDK sind threadsicher. Um eine bestmögliche Leistung zu erzielen, sollten Sie sie als langlebige Objekte behandeln. Jeder Client verfügt über seine eigene Verbindungspool-Ressource. Explizit Clients herunterfahren, wenn sie nicht mehr benötigt werden, um Ressourcenverluste zu vermeiden.

Um einen Client explizit herunterzufahren, rufen Sie die `shutdown`-Methode auf. Nach dem Aufruf von `shutdown` werden alle Client-Ressourcen freigegeben und der Client kann nicht mehr verwendet werden.

### So fahren Sie einen Client herunter

```
AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();  
ddb.shutdown();  
// Client is now unusable
```

## Geben Sie temporäre Anmeldeinformationen für AWS SDK für Java

Um Anfragen an zu stellen Amazon Web Services, müssen Sie AWS temporäre Anmeldeinformationen angeben, AWS SDK für Java die beim Aufrufen der Dienste verwendet werden können. Dafür können Sie eine der folgenden Möglichkeiten auswählen:

- Verwenden Sie die standardmäßige Anbieterkette von Anmeldeinformationen (empfohlen).
- Nutzen Sie einen bestimmten Anbieter bzw. eine Anbieterkette von Anmeldeinformationen (oder erstellen Sie Ihren eigenen).
- Geben Sie die temporären Anmeldeinformationen selbst im Code ein.

## Verwenden der standardmäßigen Anbieterkette von Anmeldeinformationen


Wenn Sie einen neuen Dienstclient ohne Angabe von Argumenten initialisieren, AWS SDK für Java versucht der Client, temporäre Anmeldeinformationen mithilfe der von der Klasse implementierten Standardanbieterkette für Anmeldeinformationen zu finden. [DefaultAWSCredentialsProviderChain](#)  
Die standardmäßige Anbieterkette von Anmeldeinformationen sucht in dieser Reihenfolge nach Anmeldeinformationen:

1. Umgebungsvariablen -AWS\_ACCESS\_KEY\_ID, AWS\_SECRET\_KEY oder AWS\_SECRET\_ACCESS\_KEY, und. AWS\_SESSION\_TOKEN Der AWS SDK für Java verwendet die [EnvironmentVariableCredentialsProvider](#) Klasse, um diese Anmeldeinformationen zu laden.
2. Java-Systemeigenschaften -aws.accessKeyId, aws.secretKey (aber nicht aws.secretAccessKey) und aws.sessionToken. Der AWS SDK für Java verwendet die [SystemPropertiesCredentialsProvider](#), um diese Anmeldeinformationen zu laden.
3. Web-Identitätstoken-Anmeldeinformationen aus der Umgebung oder dem Container.
4. Die Standarddatei mit Profilen für Anmeldeinformationen. Sie befindet sich normalerweise unter `~/.aws/credentials` (der Speicherort kann je nach Plattform variieren) und wird von vielen AWS SDKs und von der gemeinsam genutzt. AWS CLI Die AWS SDK für Java verwendet die [ProfileCredentialsProvider](#), um diese Anmeldeinformationen zu laden.

Sie können eine Datei mit den Anmeldeinformationen erstellen, indem Sie den von der bereitgestellten `aws configure` Befehl verwenden AWS CLI, oder Sie können sie erstellen, indem Sie die Datei mit einem Texteditor bearbeiten. Informationen zum Dateiformat für Anmeldeinformationen finden Sie unter [Dateiformat AWS für Anmeldeinformationen](#).

5. Amazon ECS-Container-Anmeldeinformationen — werden aus dem Amazon ECS geladen, wenn die Umgebungsvariable gesetzt `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` ist. Der AWS SDK für Java verwendet die [ContainerCredentialsProvider](#), um diese Anmeldeinformationen zu laden. Sie können die IP-Adresse für diesen Wert angeben.
6. Anmeldeinformationen für das Instanzprofil — werden auf EC2-Instances verwendet und über den Amazon EC2 Metadatendienst bereitgestellt. Der AWS SDK für Java verwendet die

[InstanceProfileCredentialsProvider](#), um diese Anmeldeinformationen zu laden. Sie können die IP-Adresse für diesen Wert angeben.

 Note

Instance-Profil-Anmeldeinformationen werden nur verwendet, wenn `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` nicht gesetzt ist. Weitere Informationen finden Sie unter [EC2ContainerCredentialsProviderWrapper](#).

## Legen Sie temporäre Anmeldeinformationen fest

Um AWS temporäre Anmeldeinformationen verwenden zu können, müssen sie an mindestens einem der oben genannten Speicherorte eingerichtet sein. Weitere Informationen über das Festlegen von Anmeldeinformationen finden Sie in den folgenden Themen:

- Informationen zum Angeben von Anmeldeinformationen in der Umgebung oder in der Standarddatei mit den Anmeldeinformationen finden Sie unter [the section called “Konfigurieren Sie temporäre Anmeldeinformationen”](#).
- Informationen über das Festlegen von Java-Systemeigenschaften finden Sie in der [System Properties](#)-Anleitung auf der offiziellen Java Tutorials-Website.
- Informationen zum Einrichten und Verwenden von Anmeldeinformationen für das Instanzprofil mit Ihren EC2-Instances finden Sie unter [Using IAM Roles to Grant Access to AWS Resources on Amazon EC2](#).

## Legen Sie ein alternatives Anmeldeinformationsprofil fest

Das AWS SDK für Java verwendet standardmäßig das Standardprofil, es gibt jedoch Möglichkeiten, anzupassen, welches Profil aus der Anmeldeinformationsdatei stammt.

Sie können die Umgebungsvariable `AWS_PROFILE` verwenden, um das vom SDK geladene Profil zu ändern.


Unter Linux, macOS oder Unix würden Sie beispielsweise den folgenden Befehl ausführen, um das Profil in `MyProfile` zu ändern.

```
export AWS_PROFILE="myProfile"
```

Verwenden Sie unter Windows folgende Variante:

```
set AWS_PROFILE="myProfile"
```

Das Festlegen der `AWS_PROFILE` Umgebungsvariablen wirkt sich auf das Laden der Anmeldeinformationen für alle offiziell unterstützten AWS SDKs und Tools aus (einschließlich AWS CLI und der). AWS Tools for Windows PowerShell Um nur das Profil für eine Java-Anwendung zu ändern, können Sie stattdessen die Systemeigenschaft `aws.profile` verwenden.

 Note

Die Umgebungsvariable hat Vorrang vor der Systemeigenschaft.

Legen Sie einen alternativen Speicherort für die Anmeldeinformationsdatei fest

Das AWS SDK für Java lädt AWS temporäre Anmeldeinformationen automatisch aus dem Standardspeicherort der Anmeldeinformationsdatei. Sie können jedoch auch den Speicherort angeben, indem Sie die Umgebungsvariable `AWS_CREDENTIAL_PROFILES_FILE` auf den vollständigen Pfad zur Anmeldeinformationsdatei setzen.

Sie können diese Funktion verwenden, um vorübergehend den Speicherort zu ändern, an dem AWS SDK für Java nach Ihrer Anmeldeinformationsdatei gesucht wird (z. B. indem Sie diese Variable in der Befehlszeile festlegen). Alternativ können Sie die Umgebungsvariable in Ihrer Benutzer- oder Systemumgebung setzen, um sie für den Benutzer oder systemweit zu ändern.

So überschreiben Sie den Standardspeicherort der Anmeldeinformationsdatei

- Setzen Sie die `AWS_CREDENTIAL_PROFILES_FILE` Umgebungsvariable auf den Speicherort Ihrer AWS Anmeldeinformationsdatei.
  - Verwenden Sie unter Linux, macOS oder Unix:

```
export AWS_CREDENTIAL_PROFILES_FILE=path/to/credentials_file
```

- Verwenden Sie unter Windows:

```
set AWS_CREDENTIAL_PROFILES_FILE=path/to/credentials_file
```

## Dateiformat für **Anmeldeinformationen**

Wenn Sie den [Anweisungen in der Grundkonfiguration](#) dieses Handbuchs folgen, sollte Ihre Anmeldeinformationsdatei das folgende grundlegende Format haben.

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

[profile2]
aws_access_key_id=<value from AWS access portal>
aws_secret_access_key=<value from AWS access portal>
aws_session_token=<value from AWS access portal>
```

Der Profilname wird in eckigen Klammern angegeben (zum Beispiel [default]), gefolgt von den konfigurierbaren Feldern in diesem Profil als Schlüssel-Wert-Paare. Ihre `credentials` Datei kann mehrere Profile enthalten, die hinzugefügt oder bearbeitet werden können, indem Sie das `aws configure --profile PROFILE_NAME` zu konfigurierende Profil auswählen.

Sie können zusätzliche Felder angeben, z. `metadata_service_timeout` B. `umetadata_service_num_attempts`. Diese können nicht mit der CLI konfiguriert werden. Sie müssen die Datei von Hand bearbeiten, wenn Sie sie verwenden möchten. Weitere Informationen zur Konfigurationsdatei und ihren verfügbaren Feldern finden Sie unter [Konfiguration von AWS Command Line Interface im](#) AWS Command Line Interface Benutzerhandbuch.

### Anmeldeinformationen laden

Nachdem Sie temporäre Anmeldeinformationen festgelegt haben, lädt das SDK sie mithilfe der standardmäßigen Anbieterkette für Anmeldeinformationen.

Dazu instanziiieren Sie wie folgt einen AWS-Service Client, ohne dem Builder explizit Anmeldeinformationen zur Verfügung zu stellen.

```
AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();
```

## Geben Sie einen Anbieter oder eine Anbieterkette für Anmeldeinformationen an

Sie können einen Anmeldeinformationsanbieter angeben, der sich von der standardmäßigen Anbieterkette von Anmeldeinformationen unterscheidet. Verwenden Sie dazu den Client-Generator.

Sie stellen einem Client Builder, der eine [AWSCredentialsProvider](#) Schnittstelle als Eingabe verwendet, eine Instanz eines Anbieters oder einer Anbieterkette für Anmeldeinformationen zur Verfügung. Das folgende Beispiel zeigt konkret, wie Sie Anmeldeinformationen der Umgebung nutzen.

```
AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
    .withCredentials(new EnvironmentVariableCredentialsProvider())
    .build();
```

Eine vollständige Liste der von AWS SDK für Java-bereitgestellten Anmeldeinformationsanbietern und Anbieterketten finden Sie unter [Alle bekannten Implementierungsklassen](#) in.

### [AWSCredentialsProvider](#)

#### Note

Sie können diese Technik verwenden, um Anmeldeinformationsanbieter oder Anbieterketten bereitzustellen, die Sie erstellen, indem Sie Ihren eigenen Anmeldeinformationsanbieter verwenden, der die `AWSCredentialsProvider` Schnittstelle implementiert, oder indem Sie der Klasse Unterklassen zuordnen. [AWSCredentialsProviderChain](#)

## Geben Sie explizit temporäre Anmeldeinformationen an

Wenn die Standard-Anmeldeinformationen oder ein bestimmter oder benutzerdefinierter Anbieter oder eine Anbieterkette für Ihren Code nicht funktionieren, können Sie explizit angegebene Anmeldeinformationen festlegen. Wenn Sie temporäre Anmeldeinformationen mit abgerufen haben AWS STS, verwenden Sie diese Methode, um die Anmeldeinformationen für den AWS Zugriff anzugeben.

1. Instanzieren Sie die [BasicSessionCredentials](#) Klasse und stellen Sie ihr den AWS Zugriffsschlüssel, den AWS geheimen Schlüssel und das AWS Sitzungstoken zur Verfügung, die das SDK für die Verbindung verwenden wird.

2. Erstellen Sie eine [AWSStaticCredentialsProvider](#) mit dem `AWSCredentials` Objekt.
3. Konfigurieren Sie den Client-Generator mit dem `AWSStaticCredentialsProvider` und erstellen Sie den Client.

Im Folgenden wird ein -Beispiel gezeigt.

```
BasicSessionCredentials awsCreds = new BasicSessionCredentials("access_key_id",
    "secret_key_id", "session_token");
AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
    .withCredentials(new AWSStaticCredentialsProvider(awsCreds))
    .build();
```

## Weitere Infos

- [Melden Sie sich an AWS und erstellen Sie einen IAM-Benutzer](#)
- [Richten Sie AWS Anmeldeinformationen und Region für die Entwicklung ein](#)
- [Verwenden von IAM-Rollen, um Zugriff auf AWS Ressourcen zu gewähren für Amazon EC2](#)

## AWS-Region Auswahl

Regionen ermöglichen Ihnen den Zugriff auf AWS Dienste, die sich physisch in einem bestimmten geografischen Gebiet befinden. Dies ist nicht nur für die Redundanz nützlich, sondern sorgt auch dafür, dass Ihre Daten und Anwendungen in der Nähe Ihres Standorts sowie des Standorts Ihrer Benutzer ausgeführt werden.

## Überprüfung der Serviceverfügbarkeit in einer Region

Um zu sehen, ob ein bestimmtes Produkt in einer Region verfügbar ist, wenden Sie die `isServiceSupported` Methode für die Region an, die Sie verwenden möchten.

```
Region.getRegion(Regions.US_WEST_2)
    .isServiceSupported(AmazonDynamoDB.ENDPOINT_PREFIX);
```

Weitere Informationen über die Regionen, die Sie angeben können, und über die Nutzung des Endpunkt-Präfixes für den abzufragenden Service finden Sie in der Dokumentation der [Regions](#)-Klasse. Jedes Service-Endpunkt-Präfix wird in der Service-Schnittstelle definiert. Das DynamoDB Endpunktpräfix ist beispielsweise in [AmazonDynamoDB](#) definiert.

## Auswählen einer Region

Ab Version 1.4 von können Sie einen Regionsnamen angeben AWS SDK für Java, und das SDK wählt automatisch einen geeigneten Endpunkt für Sie aus. Informationen darüber, wie Sie den Endpunkt selbst auswählen können, finden Sie unter [Auswahl eines bestimmten Endpunkts](#).

Um explizit eine Region festzulegen, empfehlen wir, dass Sie die [Regions](#)-Aufzählung nutzen. Dabei handelt es sich um eine Aufzählung aller öffentlich verfügbaren Regionen. Mit dem folgenden Code können Sie einen Client mit einer Region aus der Aufzählung erstellen:

```
AmazonEC2 ec2 = AmazonEC2ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();
```

Wenn die Region, die Sie verwenden möchten, nicht in der Regions-Aufzählung enthalten ist, können Sie die Region mit einem String festlegen, der den Namen der Region enthält.

```
AmazonEC2 ec2 = AmazonEC2ClientBuilder.standard()
    .withRegion("{region_api_default}")
    .build();
```

### Note

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

## Auswahl eines bestimmten Endpunkts

Jeder AWS Client kann so konfiguriert werden, dass er einen bestimmten Endpunkt innerhalb einer Region verwendet, indem die `withEndpointConfiguration` Methode bei der Erstellung des Clients aufgerufen wird.

Verwenden Sie beispielsweise den folgenden Code, um den Amazon S3 Client für die Verwendung der Region Europa (Irland) zu konfigurieren.

```
AmazonS3 s3 = AmazonS3ClientBuilder.standard()
    .withEndpointConfiguration(new EndpointConfiguration(
        "https://s3.eu-west-1.amazonaws.com",
```

```
    "eu-west-1"))  
    .withCredentials(CREDENTIALS_PROVIDER)  
    .build();
```

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

## Ermitteln Sie die Region automatisch anhand der Umgebung

### Important

Dieser Abschnitt gilt nur, wenn Sie einen [Client Builder](#) für den Zugriff auf AWS Dienste verwenden. AWS Clients, die mit dem Client-Konstruktor erstellt wurden, ermitteln die Region nicht automatisch anhand der Umgebung, sondern verwenden stattdessen die Standard-SDK-Region (useAST1).

Wenn Sie auf Amazon EC2 oder Lambda ausgeführt werden, möchten Sie möglicherweise Clients so konfigurieren, dass sie dieselbe Region verwenden, in der Ihr Code ausgeführt wird. So wird der Code von der Umgebung abgekoppelt, in der er läuft, wodurch die Bereitstellung Ihrer Anwendung in mehreren Regionen einfacher wird. Dies wiederum sorgt für weniger Latenz und mehr Redundanz.

Sie sollten Client-Generatoren verwenden, damit das SDK die Region, in der der Code ausgeführt wird, automatisch erkennt.

Verwenden Sie die `defaultClient` Methode des Client Builders, um die Region anhand der Umgebung anhand der `credential/region` Standardanbieterkette zu ermitteln.

```
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();
```

Dies entspricht der Verwendung von `standard`, gefolgt von `build`.

```
AmazonEC2 ec2 = AmazonEC2ClientBuilder.standard()  
    .build();
```

Wenn Sie nicht explizit eine Region mit den `withRegion`-Methoden festlegen, nutzt das SDK die Standard-Anbieterkette für Regionen und versucht, die zu nutzende Region zu ermitteln.

## Standard-Anbieterkette für Regionen

Folgende Regeln gelten für das Nachschlagen der Region:

1. Etwaige explizite, mit `withRegion` oder `setRegion` festgelegte Regionen direkt im Generator haben Vorrang vor allen anderen.
2. Die Umgebungsvariable `AWS_REGION` wird geprüft. Wenn sie festgelegt ist, wird die zugehörige Region zur Konfiguration des Clients verwendet.

### Note

Diese Umgebungsvariable wird vom Lambda Container festgelegt.

3. Das SDK überprüft die AWS gemeinsam genutzte Konfigurationsdatei (normalerweise unter `~/ .aws/config`). Ist die Eigenschaft `region` vorhanden, wird sie vom SDK verwendet.
  - 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 können verwendet werden, um das Profil anzupassen, das vom SDK geladen wird.
4. Das SDK versucht, mithilfe des Amazon EC2 Instanz-Metadatendienstes die Region der aktuell ausgeführten Amazon EC2 Instanz zu ermitteln.
5. Hat das SDK zu diesem Zeitpunkt immer noch keine Region gefunden, schlägt die Erstellung des Clients mit einer Ausnahme fehl.

Bei der Entwicklung von AWS Anwendungen besteht ein gängiger Ansatz darin, die gemeinsam genutzte Konfigurationsdatei (beschrieben [unter Verwenden der Standardanbieterkette für Anmeldeinformationen](#)) zu verwenden, um die Region für die lokale Entwicklung festzulegen, und sich bei der Ausführung auf der AWS Infrastruktur auf die Standardregions-Anbieterkette zu verlassen, um die Region zu bestimmen. Dies vereinfacht die Client-Erstellung stark und sorgt dafür, dass Ihre Anwendung portabel bleibt.

## Umgang mit Ausnahmen

Es ist wichtig zu verstehen, wie AWS SDK für Java und wann Ausnahmen ausgelöst werden, um qualitativ hochwertige Anwendungen mithilfe des SDK zu erstellen. In den folgenden Abschnitten

werden die verschiedenen Fälle von Ausnahmen beschrieben, die vom SDK ausgelöst werden, und wie sie korrekt verarbeitet werden.

## Warum ungeprüfte Ausnahmen?

Das AWS SDK für Java verwendet aus den folgenden Gründen Laufzeitausnahmen (oder ungeprüfte Ausnahmen) anstelle von geprüften Ausnahmen:

- 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.

Weitere Informationen über die Verwendung von geprüften und ungeprüften Ausnahmen finden Sie unter:

- [Ungeprüfte Ausnahmen — Die Kontroverse](#)
- [The Trouble with Checked Exceptions](#)
- [Java's checked exceptions were a mistake \(and here's what I would like to do about it\)](#)

## AmazonServiceException (und Unterklassen)

[AmazonServiceException](#) ist die häufigste Ausnahme, die bei der AWS SDK für Java Verwendung von auftritt. Diese Ausnahme stellt eine Fehlerantwort von einem dar AWS-Service. Wenn Sie beispielsweise versuchen, eine Amazon EC2 Instanz zu beenden, die nicht existiert, gibt EC2 eine Fehlerantwort zurück, und alle Details dieser Fehlerantwort werden in der `AmazonServiceException` ausgelösten Antwort enthalten. In einigen Fällen wird eine abgeleitete Klasse von `AmazonServiceException` ausgelöst. So erhalten Entwickler genaue Kontrolle über den Umgang mit Fehlerfällen in Catch-Blöcken.

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

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

- zurückgegebener HTTP-Statuscode
- AWS Fehlercode zurückgegeben
- detaillierte Fehlermeldung aus dem Service
- AWS Anforderungs-ID für die fehlgeschlagene Anfrage

`AmazonServiceException` enthält auch Informationen darüber, ob die fehlgeschlagene Anfrage vom Anrufer (eine Anfrage mit unzulässigen Werten) oder vom AWS-Service Anrufer (ein interner Dienstfehler) verschuldet wurde.

## AmazonClientException

[AmazonClientException](#) gibt an, dass im Java-Client-Code ein Problem aufgetreten ist, entweder beim Versuch, eine Anfrage an zu senden, AWS oder beim Versuch, eine Antwort von zu analysieren. AWS An `AmazonClientException` ist im Allgemeinen schwerwiegender als ein `AmazonServiceException` und weist auf ein schwerwiegendes Problem hin, das den Client daran hindert, Serviceanfragen an AWS Dienste zu tätigen. Dies ist beispielsweise der AWS SDK für Java Fall, `AmazonClientException` wenn keine Netzwerkverbindung verfügbar ist, wenn Sie versuchen, einen Vorgang auf einem der Clients aufzurufen.

## Asynchrone Programmierung

Sie können synchrone oder asynchrone Methoden verwenden, um Operationen für Dienste aufzurufen. AWS 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. Das AWS SDK für Java bietet zwei Möglichkeiten: Zukünftige Objekte und Callback-Methoden.

### Java-Futures

Asynchrone Methoden AWS SDK für Java geben ein [Future-Objekt](#) zurück, das die Ergebnisse der asynchronen Operation in der future enthält.

Rufen Sie die `Future isDone()`-Methode auf, um festzustellen, ob der Service bereits ein Antwortobjekt bereitgestellt hat. Wenn die Antwort bereit ist, können Sie das Antwortobjekt durch

Aufrufen der `Future get()`-Methode erhalten. Mit diesem Mechanismus können Sie regelmäßig eine Abfrage nach den Ergebnissen der asynchronen Operation durchführen, während die Anwendung an anderen Aufgaben arbeitet.

Hier ist ein Beispiel für eine asynchrone Operation, die eine Lambda Funktion aufruft und eine empfängt, `Future` die ein Objekt enthalten kann. [InvokeResult](#) Das `InvokeResult`-Objekt wird erst abgerufen, sobald `isDone()` `true` ergibt.

```
import com.amazonaws.services.lambda.AWSLambdaAsyncClient;
import com.amazonaws.services.lambda.model.InvokeRequest;
import com.amazonaws.services.lambda.model.InvokeResult;
import java.nio.ByteBuffer;
import java.util.concurrent.Future;
import java.util.concurrent.ExecutionException;

public class InvokeLambdaFunctionAsync
{
    public static void main(String[] args)
    {
        String function_name = "HelloFunction";
        String function_input = "{\"who\": \"SDK for Java\"}";

        AWSLambdaAsync lambda = AWSLambdaAsyncClientBuilder.defaultClient();
        InvokeRequest req = new InvokeRequest()
            .withFunctionName(function_name)
            .withPayload(ByteBuffer.wrap(function_input.getBytes()));

        Future<InvokeResult> future_res = lambda.invokeAsync(req);

        System.out.print("Waiting for future");
        while (future_res.isDone() == false) {
            System.out.print(".");
            try {
                Thread.sleep(1000);
            }
            catch (InterruptedException e) {
                System.err.println("\nThread.sleep() was interrupted!");
                System.exit(1);
            }
        }

        try {
            InvokeResult res = future_res.get();
        }
    }
}
```

```
        if (res.getStatusCode() == 200) {
            System.out.println("\nLambda function returned:");
            ByteBuffer response_payload = res.getPayload();
            System.out.println(new String(response_payload.array()));
        }
        else {
            System.out.format("Received a non-OK response from {AWS}: %d\n",
                res.getStatusCode());
        }
    }
}
catch (InterruptedException | ExecutionException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

System.exit(0);
}
}
```

## Asynchrone Callbacks

Neben der Verwendung des Future Java-Objekts zur Überwachung des Status asynchroner Anfragen können Sie mit dem SDK auch eine Klasse implementieren, die die Schnittstelle verwendet. [AsyncHandler](#) AsyncHandler bietet zwei Methoden, die je nachdem, wie die Anfrage abgeschlossen wurde, aufgerufen werden: onSuccess und onError.

Der wichtigste Vorteil der Rückruf-Schnittstelle besteht darin, dass Sie das Future-Objekt nicht mehr regelmäßig abfragen müssen, um zu ermitteln, wann die Anforderung abgeschlossen wurde. Stattdessen kann Ihr Code sofort die nächste Aktivität beginnen und sich auf das SDK verlassen, das für den Aufruf Ihrer Handler zum richtigen Zeitpunkt sorgt.

```
import com.amazonaws.services.lambda.AWSLambdaAsync;
import com.amazonaws.services.lambda.AWSLambdaAsyncClientBuilder;
import com.amazonaws.services.lambda.model.InvokeRequest;
import com.amazonaws.services.lambda.model.InvokeResult;
import com.amazonaws.handlers.AsyncHandler;
import java.nio.ByteBuffer;
import java.util.concurrent.Future;

public class InvokeLambdaFunctionCallback
{
```

```
private class AsyncLambdaHandler implements AsyncHandler<InvokeRequest,
InvokeResult>
{
    public void onSuccess(InvokeRequest req, InvokeResult res) {
        System.out.println("\nLambda function returned:");
        ByteBuffer response_payload = res.getPayload();
        System.out.println(new String(response_payload.array()));
        System.exit(0);
    }

    public void onError(Exception e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void main(String[] args)
{
    String function_name = "HelloFunction";
    String function_input = "{\\"who\\":\\"SDK for Java\\"}";

    AWSLambdaAsync lambda = AWSLambdaAsyncClientBuilder.defaultClient();
    InvokeRequest req = new InvokeRequest()
        .withFunctionName(function_name)
        .withPayload(ByteBuffer.wrap(function_input.getBytes()));

    Future<InvokeResult> future_res = lambda.invokeAsync(req, new
AsyncLambdaHandler());

    System.out.print("Waiting for async callback");
    while (!future_res.isDone() && !future_res.isCancelled()) {
        // perform some other tasks...
        try {
            Thread.sleep(1000);
        }
        catch (InterruptedException e) {
            System.err.println("Thread.sleep() was interrupted!");
            System.exit(0);
        }
        System.out.print(".");
    }
}
}
```

## Bewährte Methoden

### Ausführen eines Rückrufs

Ihre Implementierung des `AsyncHandler` wird in dem Threadpool ausgeführt, der dem asynchronen Client gehört. Kurzer, schnell ausgeführter Code eignet sich am besten für Ihre `AsyncHandler` Implementierung. Long-running oder das Blockieren von Code in Ihren Handler-Methoden kann zu Konflikten im Thread-Pool führen, der vom asynchronen Client verwendet wird, und den Client daran hindern, Anfragen auszuführen. Wenn Sie eine lang andauernde Aufgabe haben, die mit einem Rückruf gestartet werden soll, lassen Sie den Rückruf die Aufgabe in einem neuen Thread oder in einem Threadpool ausführen, der von Ihrer Anwendung verwaltet wird.

### Threadpool-Konfiguration

Die asynchronen Clients im AWS SDK für Java stellen einen Standard-Thread-Pool bereit, der für die meisten Anwendungen funktionieren sollte. Sie können einen benutzerdefinierten Code implementieren [ExecutorService](#) und ihn an AWS SDK für Java asynchrone Clients übergeben, um mehr Kontrolle darüber zu haben, wie die Thread-Pools verwaltet werden.

Sie könnten beispielsweise eine `ExecutorService` Implementierung bereitstellen, die eine benutzerdefinierte Methode verwendet, [ThreadFactory](#) um zu steuern, wie Threads im Pool benannt werden, oder um zusätzliche Informationen über die Threadnutzung zu protokollieren.

### Asynchroner Zugriff

Die [TransferManager](#) Klasse im SDK bietet asynchrone Unterstützung für die Arbeit mit. Amazon `S3TransferManager` verwaltet asynchrone Uploads und Downloads, bietet detaillierte Fortschrittsberichte zu Übertragungen und unterstützt Rückrufe bei verschiedenen Ereignissen.

## Protokollierung AWS SDK für Java Calls

Das AWS SDK für Java ist mit [Apache Commons Logging](#) instrumentiert, einer Abstraktionsschicht, die die Verwendung eines von mehreren Logging-Systemen zur Laufzeit ermöglicht.

Unterstützte Protokollierungssysteme sind u. a. das Java Logging Framework und Apache Log4j. In diesem Thema erhalten Sie Informationen zur Nutzung von Log4j. Sie können die Protokollierungsfunktionalität des SDKs ohne Änderungen am Code Ihrer Anwendung nutzen.

Weitere Informationen über [Log4j](#) finden Sie auf der [Apache-Website](#).

**Note**

In diesem Thema geht es um Log4j 1.x. Log4j2 unterstützt Apache Commons Logging nicht direkt. Stattdessen wird ein Adapter bereitgestellt, der Protokollierungsaufrufe automatisch mithilfe der Apache Commons Logging-Schnittstelle an Log4j2 weiterleitet. Weitere Informationen finden Sie unter [Commons Logging Bridge](#) in der Log4j2-Dokumentation.

## Herunterladen der Log4J-JAR

Zur Nutzung von Log4j mit dem SDK müssen Sie das Log4j-JAR von der Apache-Website herunterladen. Das SDK enthält das JAR nicht. Kopieren Sie die JAR-Datei an einen Speicherort, der in Ihrem Klassenpfad enthalten ist.

Log4j verwendet eine Konfigurationsdatei namens "log4j.properties". Beispiel-Konfigurationsdateien werden nachfolgend angezeigt. Kopieren Sie die Konfigurationsdatei in ein Verzeichnis in Ihrem Klassenpfad. Die Log4j JAR-Dateien und die Datei "log4j.properties" müssen nicht im selben Verzeichnis liegen.

In der Konfigurationsdatei "log4j.properties" sind Eigenschaften wie die [Protokollierungsebene](#), das Ziel der Protokollierungsausgaben (z. B. [an eine Datei oder an die Konsole](#)) sowie das [Ausgabeformat](#) angegeben. Die Protokollierungsebene ist die Granularität der Ausgaben, die der Protokollierer erzeugt. Log4j unterstützt das Konzept mehrerer Hierarchien der Protokollierung. Die Protokollierungsebene wird für jede Hierarchie separat festgelegt. Die folgenden zwei Protokollierungshierarchien sind im AWS SDK für Java verfügbar:

- log4j.logger.com.amazonaws
- log4j.logger.org.apache.http.wire

## Festlegen des Klassenpfads

Sowohl die Log4j JAR-Datei als auch die Datei "log4j.properties" müssen in Ihrem Klassenpfad liegen. Wenn Sie [Apache Ant](#) verwenden, legen Sie den Klassenpfad im path-Element der Ant-Datei fest. Das folgende Beispiel zeigt ein Pfadelement aus der Ant-Datei für das im Amazon S3 [SDK enthaltene Beispiel](#).

```
<path id="aws.java.sdk.classpath">
  <fileset dir="../../third-party" includes="**/*.jar"/>
```

```
<fileset dir="../../lib" includes="**/*.jar"/>
<pathelement location="."/>
</path>
```

In der Eclipse-IDE können Sie den Klassenpfad festlegen, indem Sie das Menü öffnen und auf **Projekt | Eigenschaften | Java Build-Pfad** klicken.

## Service-Specific Fehler und Warnungen

Wir empfehlen, dass Sie die Protokollierungshierarchie "com.amazonaws" immer auf "WARN" gestellt lassen. So entgehen Ihnen keine wichtigen Meldungen aus den Client-Bibliotheken. Wenn der Amazon S3 Client beispielsweise feststellt, dass Ihre Anwendung nicht ordnungsgemäß geschlossen wurde, InputStream und möglicherweise Ressourcen verloren gehen, meldet der S3-Client dies in Form einer Warnmeldung an die Protokolle. Dadurch wird auch sichergestellt, dass Nachrichten protokolliert werden, wenn der Client Schwierigkeiten bei der Verarbeitung von Anforderungen oder Antworten hat.

In der folgenden "log4j.properties"-Datei ist der rootLogger auf WARN gesetzt. Dies hat zur Folge, dass Warn- und Fehlermeldungen von allen Protokollierern in der Hierarchie "com.amazonaws" enthalten sind. Alternativ können Sie ausdrücklich den com.amazonaws-Protokollierer auf WARN stellen.

```
log4j.rootLogger=WARN, A1
log4j.appender.A1=org.apache.log4j.ConsoleAppender
log4j.appender.A1.layout=org.apache.log4j.PatternLayout
log4j.appender.A1.layout.ConversionPattern=%d [%t] %-5p %c - %m%n
# Or you can explicitly enable WARN and ERROR messages for the {AWS} Java clients
log4j.logger.com.amazonaws=WARN
```

## Request/Response Protokollierung im Überblick

Jede Anfrage an eine AWS-Service generiert eine eindeutige AWS Anforderungs-ID, die nützlich ist, wenn Sie auf ein Problem mit der Bearbeitung einer Anfrage stoßen. AWS-Service AWS Anfrage-IDs sind programmgesteuert über Ausnahmeobjekte im SDK für jeden fehlgeschlagenen Serviceabruf zugänglich. Sie können auch über die DEBUG-Protokollebene im Logger „com.amazonaws.request“ gemeldet werden.

Die folgende Datei log4j.properties ermöglicht eine Zusammenfassung der Anfragen und Antworten, einschließlich der Anfrage-IDs. AWS

```
log4j.rootLogger=WARN, A1
log4j.appender.A1=org.apache.log4j.ConsoleAppender
log4j.appender.A1.layout=org.apache.log4j.PatternLayout
log4j.appender.A1.layout.ConversionPattern=%d [%t] %-5p %c - %m%n
# Turn on DEBUG logging in com.amazonaws.request to log
# a summary of requests/responses with {AWS} request IDs
log4j.logger.com.amazonaws.request=DEBUG
```

Hier finden Sie ein Beispiel für die Protokollausgabe.

```
2009-12-17 09:53:04,269 [main] DEBUG com.amazonaws.request - Sending
Request: POST https://rds.amazonaws.com / Parameters: (MaxRecords: 20,
Action: DescribeEngineDefaultParameters, SignatureMethod: HmacSHA256,
AWSAccessKeyId: ACCESSKEYID, Version: 2009-10-16, SignatureVersion: 2,
Engine: mysql5.1, Timestamp: 2009-12-17T17:53:04.267Z, Signature:
q963XH63Lcovl5Rr71APlzlye99rmWwT9DfuQaNznkD, ) 2009-12-17 09:53:04,464
[main] DEBUG com.amazonaws.request - Received successful response: 200, {AWS}
Request ID: 694d1242-cee0-c85e-f31f-5dab1ea18bc6 2009-12-17 09:53:04,469
[main] DEBUG com.amazonaws.request - Sending Request: POST
https://rds.amazonaws.com / Parameters: (ResetAllParameters: true, Action:
ResetDBParameterGroup, SignatureMethod: HmacSHA256, DBParameterGroupName:
java-integ-test-param-group-00000000000000, AWSAccessKeyId: ACCESSKEYID,
Version: 2009-10-16, SignatureVersion: 2, Timestamp:
2009-12-17T17:53:04.467Z, Signature:
9WcgfPwTobvLVcpyhbrdN7P713uH0oviYQ4yZ+TQjsQ=, )

2009-12-17 09:53:04,646 [main] DEBUG com.amazonaws.request - Received
successful response: 200, {AWS} Request ID:
694d1242-cee0-c85e-f31f-5dab1ea18bc6
```

## Verbose-Protokollierung des Netzwerkverkehrs

In einigen Fällen kann es nützlich sein, die genauen Anfragen und Antworten zu sehen, die der Sender AWS SDK für Java sendet und empfängt. Sie sollten diese Protokollierung in Produktionssystemen nicht aktivieren, da das Ausschreiben umfangreicher Anfragen (z. B. wenn eine Datei hochgeladen wird Amazon S3) oder Antworten eine Anwendung erheblich verlangsamen kann. Wenn Sie wirklich Zugriff auf diese Informationen benötigen, können Sie sie vorübergehend über den Apache HttpClient 4-Logger aktivieren. Durch Aktivieren der DEBUG-Ebene für den `org.apache.http.wire`-Protokollierer wird die Protokollierung für sämtliche Anforderungs- und Antwortdaten aktiviert.

Die folgende Datei `log4j.properties` aktiviert die vollständige Protokollierung in Apache HttpClient 4 und sollte nur vorübergehend aktiviert werden, da dies erhebliche Auswirkungen auf die Leistung Ihrer Anwendung haben kann.

```
log4j.rootLogger=WARN, A1
log4j.appender.A1=org.apache.log4j.ConsoleAppender
log4j.appender.A1.layout=org.apache.log4j.PatternLayout
log4j.appender.A1.layout.ConversionPattern=%d [%t] %-5p %c - %m%n
# Log all HTTP content (headers, parameters, content, etc) for
# all requests and responses. Use caution with this since it can
# be very expensive to log such verbose data!
log4j.logger.org.apache.http.wire=DEBUG
```

## Protokollieren von Latenz-Metriken

Wenn Sie bei der Behandlung Metriken anzeigen möchten, z. B. welcher Prozess die meiste Zeit beansprucht oder ob die Latenz auf der Server- oder Client-Seite größer ist, kann der Latenz-Protokollierer hilfreich sein. Stellen Sie den `com.amazonaws.latency`-Protokollierer zur Aktivierung auf `DEBUG`.

### Note

Dieser Protokollierer ist nur verfügbar, wenn SDK-Metriken aktiviert sind. Weitere Informationen zum SDK-Metrikenpaket finden Sie unter [Metriken aktivieren für AWS SDK für Java](#)

```
log4j.rootLogger=WARN, A1
log4j.appender.A1=org.apache.log4j.ConsoleAppender
log4j.appender.A1.layout=org.apache.log4j.PatternLayout
log4j.appender.A1.layout.ConversionPattern=%d [%t] %-5p %c - %m%n
log4j.logger.com.amazonaws.latency=DEBUG
```

Hier finden Sie ein Beispiel für die Protokollausgabe.

```
com.amazonaws.latency - ServiceName=[{S3}], StatusCode=[200],
ServiceEndpoint=[https://list-objects-integ-test-test.s3.amazonaws.com],
RequestType=[ListObjectsV2Request], AWSRequestID=[REQUESTID],
HttpClientPoolPendingCount=0,
```

```
RetryCapacityConsumed=0, HttpClientPoolAvailableCount=0, RequestCount=1,  
HttpClientPoolLeasedCount=0, ResponseProcessingTime=[52.154],  
ClientExecuteTime=[487.041],  
HttpClientSendRequestTime=[192.931], HttpRequestTime=[431.652],  
RequestSigningTime=[0.357],  
CredentialsRequestTime=[0.011, 0.001], HttpClientReceiveResponseTime=[146.272]
```

## Client-Konfiguration

Der AWS SDK für Java ermöglicht es Ihnen, die Standard-Client-Konfiguration zu ändern. Dies ist hilfreich, wenn Sie:

- Herstellen einer Internetverbindung über einen Proxy
- Ändern von HTTP-Transport-Einstellungen, z. B. Verbindungstimeout und wiederholte Anforderungsversuche
- Angabe von TCP-Socketpuffer-Größenhinweisen

## Proxy-Konfiguration

Wenn Sie ein Client-Objekt erstellen, können Sie ein optionales [ClientConfiguration](#) Objekt übergeben, um die Konfiguration des Clients anzupassen.

Wenn Sie über einen Proxyserver eine Verbindung zum Internet herstellen, müssen Sie Ihre Proxyserver-Einstellungen (Proxyhost, Port und username/password) über das `ClientConfiguration` Objekt konfigurieren.

## HTTP-Transport-Konfiguration

Mithilfe des [ClientConfiguration](#) Objekts können Sie mehrere HTTP-Transportoptionen konfigurieren. Gelegentlich werden neue Optionen hinzugefügt. Eine vollständige Liste der Optionen, die Sie abrufen oder festlegen können, finden Sie in der AWS SDK für Java API-Referenz.

### Note

Jeder konfigurierbare Wert hat einen von einer Konstanten definierten Standardwert. Eine Liste der konstanten Werte für finden Sie `ClientConfiguration` unter [Konstante Feldwerte](#) in der AWS SDK für Java API-Referenz.

## Maximale Anzahl der Verbindungen

Sie können die maximal zulässige Anzahl offener HTTP-Verbindungen mit der Methode [ClientConfiguration.setMaxConnections](#) festlegen.

### Important

Legen Sie die maximalen Verbindungen auf die Anzahl der gleichzeitigen Transaktionen fest, um Verbindungskonflikte und eine schlechte Leistung zu vermeiden. Den Standardwert für maximale Verbindungen finden Sie unter [Konstante Feldwerte](#) in der AWS SDK für Java API-Referenz.

## Timeouts und Fehlerbehandlung

Sie können Optionen im Zusammenhang mit Timeouts und der Fehlerbehandlung für HTTP-Verbindungen festlegen.

- Verbindungstimeout

Das Verbindungstimeout ist die Zeit (in Millisekunden), die die HTTP-Verbindung wartet, um eine Verbindung herzustellen. Der Standardwert beträgt 10 000 ms.

Verwenden Sie die [ClientConfiguration.setConnectionTimeout](#) Methode, um diesen Wert selbst festzulegen.

- Connection Time to Live (TTL, Gültigkeitsdauer der Verbindung)

Standardmäßig versucht das SDK, HTTP-Verbindungen so lange wie möglich wiederzuverwenden. In Fehlersituationen, bei denen eine Verbindung zu einem Server aufgebaut wird, der außer Betrieb genommen wird, sorgt eine endliche TTL für eine schnellere Anwendungswiederherstellung. Wird beispielsweise eine 15-Minuten-TTL eingestellt, ist dadurch sichergestellt, dass auch dann, wenn Sie eine Verbindung zu einem Server aufgebaut haben, bei dem Probleme auftreten, innerhalb von 15 Minuten eine Verbindung zu einem neuen Server hergestellt wird.

Verwenden Sie die [ClientConfiguration.setConnectionTTL](#) Methode, um die TTL der HTTP-Verbindung festzulegen.

- Maximale Anzahl der erneuten Versuche

Die standardmäßige maximale Wiederholungsanzahl für Fehler bei wiederholbaren Aktionen ist 3. Mit der [ClientConfiguration.setMaxErrorRetry](#) Methode können Sie einen anderen Wert festlegen.

## Lokale Adresse

Um die lokale Adresse festzulegen, an die der HTTP-Client gebunden wird, verwenden Sie [ClientConfiguration.setLocalAddress](#).

## TCP-Socketpuffer-Größenhinweise

Fortgeschrittene Benutzer, die TCP-Parameter auf niedriger Ebene optimieren möchten, können zusätzlich Hinweise zur Größe des TCP-Puffers über das [ClientConfiguration](#) Objekt festlegen. Die meisten Benutzer müssen diese Einstellungen nie anpassen, sie stehen aber für fortgeschrittene Benutzer bereit.

Die optimalen TCP-Puffergrößen für eine Anwendung hängen stark von der Konfiguration und den Fähigkeiten des Netzwerks und des Betriebssystems ab. Beispielsweise bieten die meisten modernen Betriebssysteme eine Logik zur automatischen Anpassung der TCP-Puffergrößen. Dies kann sich weitreichend auf die Leistung von TCP-Verbindungen auswirken, wenn diese lang genug geöffnet bleiben, damit die automatische Anpassung die Puffergrößen optimieren kann.

Große Puffer (z. B. 2 MB) ermöglichen dem Betriebssystem, mehr Daten im Arbeitsspeicher zu puffern, ohne dass der Remote-Server die Informationen bestätigen muss. Sie sind also besonders nützlich, wenn das Netzwerk eine hohe Latenz aufweist.

Dies ist nur ein Hinweis, an den sich das Betriebssystem nicht halten muss. Bei Verwendung dieser Option sollten Benutzer immer die beim Betriebssystem konfigurierten Grenz- und Standardwerte überprüfen. In den meisten Betriebssystemen ist eine maximale TCP-Puffergröße konfiguriert, die nicht überschritten wird, es sei denn, Sie heben die maximale TCP-Puffergröße explizit an.

Für die Konfiguration von TCP-Puffergrößen und betriebssystemspezifischen TCP-Einstellungen stehen viele Ressourcen zur Verfügung, wie z. B.:

- [Host Tuning](#)

## Zugriffskontrollrichtlinien

AWS Mithilfe von Zugriffskontrollrichtlinien können Sie detaillierte Zugriffskontrollen für Ihre Ressourcen festlegen. Eine Zugriffskontrollrichtlinie besteht aus einer Reihe von Anweisungen, die folgende Form annehmen:

Konto A darf Aktion B auf Ressource C ausführen, wenn Bedingung D gilt.

Wobei Folgendes gilt:

- A ist der Principal — AWS-Konto Derjenige, der eine Anfrage zum Zugriff auf oder zur Änderung einer Ihrer AWS Ressourcen stellt.
- B ist die Aktion — die Art und Weise, wie auf Ihre AWS Ressource zugegriffen oder sie geändert wird, z. B. das Senden einer Nachricht an eine Amazon SQS Warteschlange oder das Speichern eines Objekts in einem Amazon S3 Bucket.
- C ist die Ressource — die AWS Entität, auf die der Principal zugreifen möchte, z. B. eine Amazon SQS Warteschlange oder ein Objekt, in dem gespeichert ist Amazon S3.
- D ist eine Reihe von Bedingungen — Die optionalen Einschränkungen, die angeben, wann dem Prinzipal der Zugriff auf Ihre Ressource gewährt oder verweigert werden soll. Viele ausdrucksstarke Bedingungen sind verfügbar, einige speziell für jeden Service. Beispielsweise können Sie mit Datsbedingungen den Zugriff auf Ressourcen nur nach oder vor einem bestimmten Zeitpunkt zulassen.

## Amazon S3 Beispiel

Das folgende Beispiel zeigt eine Richtlinie, die jedem Zugriff erlaubt, alle Objekte in einem Bucket zu lesen, aber den Zugriff auf das Hochladen von Objekten in diesen Bucket auf zwei bestimmte AWS-Konten beschränkt (zusätzlich zum Konto des Bucket-Besitzers).

```
Statement allowPublicReadStatement = new Statement(Effect.Allow)
    .withPrincipals(Principal.AllUsers)
    .withActions(S3Actions.GetObject)
    .withResources(new S3ObjectResource(myBucketName, "*"));
Statement allowRestrictedWriteStatement = new Statement(Effect.Allow)
    .withPrincipals(new Principal("123456789"), new Principal("876543210"))
    .withActions(S3Actions.PutObject)
    .withResources(new S3ObjectResource(myBucketName, "*"));

Policy policy = new Policy()
```

```
.withStatements(allowPublicReadStatement, allowRestrictedWriteStatement);
```

```
AmazonS3 s3 = AmazonS3ClientBuilder.defaultClient();  
s3.setBucketPolicy(myBucketName, policy.toJson());
```

## Amazon SQS Beispiel

Richtlinien werden häufig verwendet, um eine Amazon SQS Warteschlange für den Empfang von Nachrichten von einem Amazon SNS SNS-Thema zu autorisieren.

```
Policy policy = new Policy().withStatements(  
    new Statement(Effect.Allow)  
        .withPrincipals(Principal.AllUsers)  
        .withActions(SQSActions.SendMessage)  
        .withConditions(ConditionFactory.newSourceArnCondition(myTopicArn)));
```

```
Map queueAttributes = new HashMap();  
queueAttributes.put(QueueAttributeName.Policy.toString(), policy.toJson());
```

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();  
sqs.setQueueAttributes(new SetQueueAttributesRequest(myQueueUrl, queueAttributes));
```

## Beispiel für Amazon SNS

Einige Dienste bieten zusätzliche Bedingungen, die in Richtlinien verwendet werden können. Amazon SNS bietet Bedingungen für das Zulassen oder Ablehnen von Abonnements für SNS-Themen auf der Grundlage des Protokolls (z. B. E-Mail, HTTP, HTTPS Amazon SQS) und des Endpunkts (z. B. E-Mail-Adresse, URL, Amazon SQS ARN) der Anfrage zum Abonnieren eines Themas.

```
Condition endpointCondition =  
    SNSConditionFactory.newEndpointCondition("*@mycompany.com");
```

```
Policy policy = new Policy().withStatements(  
    new Statement(Effect.Allow)  
        .withPrincipals(Principal.AllUsers)  
        .withActions(SNSActions.Subscribe)  
        .withConditions(endpointCondition));
```

```
AmazonSNS sns = AmazonSNSClientBuilder.defaultClient();  
sns.setTopicAttributes(  
    new SetTopicAttributesRequest(myTopicArn, "Policy", policy.toJson()));
```

## Legen Sie die JVM-TTL für die Suche nach DNS-Namen fest

Die Java Virtual Machine (JVM) speichert DNS-Namensauflösungen zwischen. Wenn die JVM einen Hostnamen zu einer IP-Adresse auflöst, speichert sie die IP-Adresse für einen bestimmten Zeitraum zwischen. Diese Zeit ist als Time-to-Live (TTL, Lebensdauer) bekannt.

Da AWS Ressourcen DNS-Namenseinträge verwenden, die sich gelegentlich ändern, empfehlen wir Ihnen, Ihre JVM mit einem TTL-Wert von 5 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 also die IP-Adresse einer AWS Ressource ändert, während Ihre Anwendung noch läuft, 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.

### Wie legt man die JVM-TTL fest

Um die TTL der JVM zu ändern, legen Sie den Sicherheitseigenschaftswert [networkaddress.cache.ttl](#) fest. Beachten Sie, dass `networkaddress.cache.ttl` es sich um eine Sicherheitseigenschaft und nicht um eine Systemeigenschaft handelt, d. h. sie kann nicht mit dem Befehlszeilen-Flag gesetzt werden. -D

#### Option 1: Stellen Sie sie programmgesteuert in Ihrer Anwendung ein

Rufen Sie [java.security.Security.setProperty\(\)](#) früh beim Start Ihrer Anwendung an, bevor AWS SDK-Clients erstellt werden und bevor Netzwerkanforderungen gestellt werden:

```
import java.security.Security;

public class MyApplication {
    public static void main(String[] args) {
        Security.setProperty("networkaddress.cache.ttl", "5");

        // ... create SDK clients and run application
    }
}
```

## Option 2: Legen Sie es in der Datei `java.security` fest

Legen Sie die `networkaddress.cache.ttl` Eigenschaft in der `$JAVA_HOME/jre/lib/security/java.security` Datei für Java 8 oder der `$JAVA_HOME/conf/security/java.security` Datei für Java 11 oder höher fest.

Das Folgende ist ein Ausschnitt aus einer `java.security` Datei, die zeigt, dass der TTL-Cache auf 5 Sekunden eingestellt ist.

```
#  
# The Java-level namelookup cache policy for successful lookups:  
#  
# any negative value: caching forever  
# any positive value: the number of seconds to cache an address for  
# zero: do not cache  
#  
...  
networkaddress.cache.ttl=5  
...
```

Alle Anwendungen, die auf der durch die `$JAVA_HOME` Umgebungsvariable repräsentierten JVM ausgeführt werden, verwenden diese Einstellung.

## Option 3: Verwenden Sie das Fallback für JDK-Systemeigenschaften (Befehlszeile)

Wenn Sie die Sicherheitskonfiguration oder den Sicherheitscode nicht ändern können, können Sie die JDK-Systemeigenschaften verwenden. Diese dienen als Fallbacks, wenn keine Sicherheitseigenschaft definiert ist.

- `sun.net.inetaddr.ttl`— Steuert erfolgreiche Suchvorgänge (positive TTL)
- `sun.net.inetaddr.negative.ttl`— Steuert fehlgeschlagene Suchvorgänge (negative TTL)

```
java -Dsun.net.inetaddr.ttl=5 -Dsun.net.inetaddr.negative.ttl=1 -jar myapp.jar
```

### Note

Dies sind JDK-internal Eigenschaften, die in der [Oracle Java 8 Networking Properties Reference](#) als private Eigenschaften dokumentiert sind und „in future Versionen

möglicherweise nicht unterstützt werden“. Verwenden Sie nach Möglichkeit die Optionen 1—2.

## Aktivierung von Metriken für AWS SDK für Java

AWS SDK für Java Sie können mit [Amazon](#) Metriken für die Visualisierung und Überwachung generieren CloudWatch, die Folgendes messen:

- die Leistung Ihrer Anwendung beim Zugriff AWS
- die Leistung Ihrer JVMs bei Verwendung mit AWS
- Details der Laufzeitumgebung wie z. B. Heap-Speicher, Anzahl der Threads und geöffneter Datei-Deskriptoren

## Wie aktiviert man die Generierung von Java-SDK-Metriken

Sie müssen die folgende Maven-Abhängigkeit hinzufügen, damit das SDK Metriken an CloudWatch senden kann.

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-bom</artifactId>
      <version>1.12.490* </version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-cloudwatchmetrics</artifactId>
    <scope>provided</scope>
  </dependency>
  <!-- Other SDK dependencies. -->
</dependencies>
```

\* Ersetzen Sie die Versionsnummer durch die neueste Version des SDK, die bei [Maven Central](#) verfügbar ist.

AWS SDK für Java Metriken sind standardmäßig deaktiviert. Um es für Ihre lokale Entwicklungsumgebung zu aktivieren, fügen Sie eine Systemeigenschaft hinzu, die beim Start der JVM auf Ihre AWS Sicherheitsanmeldedatei verweist. Beispiel:

```
-Dcom.amazonaws.sdk.enableDefaultMetrics=credentialFile=/path/aws.properties
```

Sie müssen den Pfad zu Ihrer Anmeldeinformationsdatei angeben, damit das SDK die gesammelten Datenpunkte zur späteren Analyse hochladen kann. CloudWatch

#### Note

Wenn Sie AWS von einer Amazon EC2 Instanz aus zugreifen, die den Amazon EC2 Instanz-Metadatendienst verwendet, müssen Sie keine Anmeldeinformationsdatei angeben. In diesem Fall ist nur Folgendes anzugeben:

```
-Dcom.amazonaws.sdk.enableDefaultMetrics
```

Alle von der erfassten Metriken AWS SDK für Java befinden sich im Namespace AWSSDK/Java und werden in die CloudWatch Standardregion (us-east-1) hochgeladen. Wenn Sie die Region ändern möchten, geben Sie sie mit dem Attribut `cloudwatchRegion` in der Systemeigenschaft an. Um die CloudWatch Region beispielsweise auf us-east-1 festzulegen, verwenden Sie:

```
-Dcom.amazonaws.sdk.enableDefaultMetrics=credentialFile=/path/  
aws.properties,cloudwatchRegion={region_api_default}
```

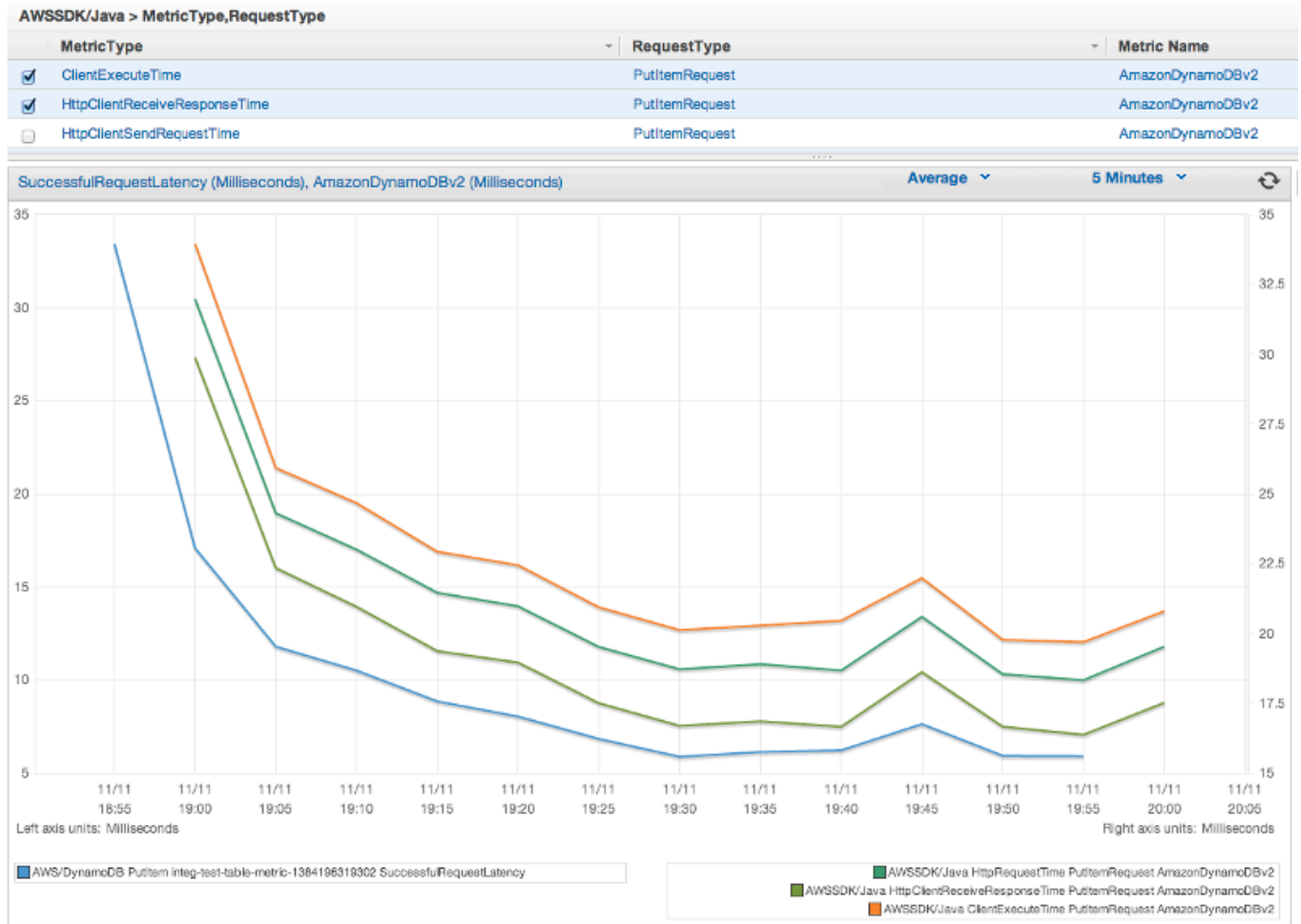
Sobald Sie die Funktion aktiviert haben, werden bei jeder Serviceanfrage AWS von metrische Datenpunkte generiert AWS SDK für Java, für die statistische Zusammenfassung in die Warteschlange gestellt und asynchron hochgeladen, CloudWatch etwa einmal pro Minute. Sobald die Metriken hochgeladen wurden, können Sie sie mithilfe von [visualisieren AWS-Managementkonsole](#) und Alarme für potenzielle Probleme wie Speicherverlust, Verlust von Dateideskriptoren usw. einrichten.

## Verfügbare Arten von Metriken

Die Standardmetriken werden in drei Hauptkategorien unterteilt:

## AWS Metriken anfordern

- Deckt Bereiche wie die Latenz des HTTP request/response, die Anzahl der Anfragen, Ausnahmen und Wiederholungen ab.



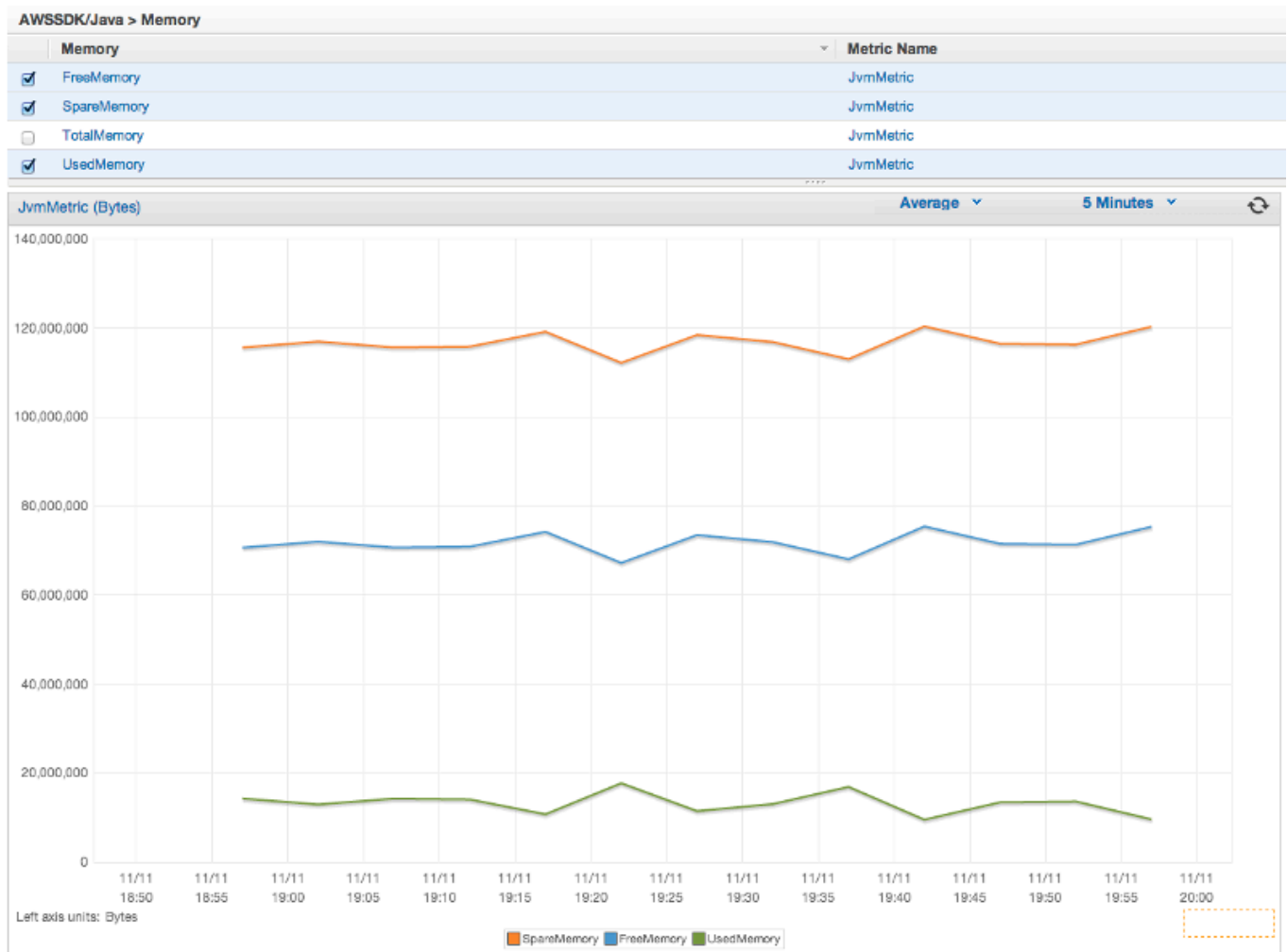
## AWS-Service Metriken

- Schließen Sie AWS-Service spezifische Daten ein, z. B. den Durchsatz und die Byteanzahl für S3-Uploads und -Downloads.



## Maschinenmetriken

- Enthalten die Laufzeitumgebung, darunter Heap-Speicher, Anzahl der Threads und geöffneter Datei-Deskriptoren.



Wenn Sie Maschinenmetriken ausschließen möchten, fügen Sie `excludeMachineMetrics` in die Systemeigenschaft ein:

```
-Dcom.amazonaws.sdk.enableDefaultMetrics=credentialFile=/path/
aws.properties,excludeMachineMetrics
```

## Weitere Informationen

- Eine vollständige Liste der vordefinierten Kernmetriktypen finden Sie in der [amazonaws/metrics Paketübersicht](#).
- Weitere Informationen zur CloudWatch Verwendung von finden Sie AWS SDK für Java in [CloudWatch Beispielen mit der AWS SDK für Java](#).

- Weitere Informationen zur Leistungsoptimierung finden Sie im Blogbeitrag [Tuning the AWS SDK für Java to Improve Resiliency](#).

# AWS SDK für Java Codebeispiele

Dieser Abschnitt enthält Tutorials und Beispiele für die Verwendung von AWS SDK für Java Version 1 zum Programmieren von AWS Diensten.

Den Quellcode für diese und andere Beispiele finden Sie im [Repository für AWS Dokumentationscodebeispiele unter GitHub](#).

Um dem AWS Dokumentationsteam ein neues Codebeispiel vorzuschlagen, das es erstellen könnte, erstellen Sie eine neue Anfrage. Das Team möchte Codebeispiele erstellen, die breitere Szenarien und Anwendungsfälle abdecken, im Vergleich zu einfachen Codeausschnitten, die nur einzelne API-Aufrufe abdecken. Anweisungen finden Sie in den [Richtlinien](#) für Beiträge im Codebeispiel-Repository auf.. GitHub

## AWS SDK für Java 2.x

Im Jahr 2018 AWS veröffentlichte das [AWS SDK for Java 2.x](#). Dieses Handbuch enthält Anweisungen zur Verwendung des neuesten Java-SDK sowie Beispielcode.

### Note

Weitere Beispiele [und zusätzliche Ressourcen für AWS SDK für Java Entwickler finden Sie unter Zusätzliche Dokumentation](#) und Ressourcen!

## CloudWatch Beispiele für die Verwendung der AWS SDK für Java

Dieser Abschnitt bietet Beispiele für die Programmierung von [CloudWatch](#) mithilfe des [AWS SDK für Java](#).

Amazon CloudWatch überwacht Ihre Amazon Web Services (AWS) Ressourcen und die Anwendungen, auf denen Sie laufen, AWS in Echtzeit. Sie können CloudWatch damit Metriken sammeln und verfolgen. Dabei handelt es sich um Variablen, die Sie für Ihre Ressourcen und Anwendungen messen können. CloudWatch Alarme senden Benachrichtigungen oder nehmen auf der Grundlage von von Ihnen festgelegter Regeln automatisch Änderungen an den Ressourcen vor, die Sie überwachen.

Weitere Informationen dazu CloudWatch finden Sie im [Amazon CloudWatch Benutzerhandbuch](#).

### Note

Die Beispiele enthalten nur den Code, der zur Demonstration jeder Technik nötig ist. Der [vollständige Beispielformatcode 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

- [Metriken abrufen von CloudWatch](#)
- [Veröffentlichen benutzerdefinierter Metrikdaten](#)
- [Mit CloudWatch Alarmen arbeiten](#)
- [Verwenden von Alarmaktionen in CloudWatch](#)
- [Ereignisse senden an CloudWatch](#)

## Metriken abrufen von CloudWatch

### Auflisten von Metriken

Um CloudWatch Metriken aufzulisten, erstellen Sie eine `listMetrics` Methode [ListMetricsRequest](#) und rufen sie auf. `AmazonCloudWatchClient` Sie können `ListMetricsRequest` zum Filtern der zurückgegebenen Metriken nach Namespace, Metrikname oder Dimensionen verwenden.

### Note

Eine Liste der Metriken und Dimensionen, die von AWS Services veröffentlicht werden, finden Sie im <https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring-cw-support-for-AWS.html> [Amazon Metrics and Dimensions Reference] im Benutzerhandbuch. CloudWatch Amazon CloudWatch

## Importe

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatch;
```

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatchClientBuilder;
import com.amazonaws.services.cloudwatch.model.ListMetricsRequest;
import com.amazonaws.services.cloudwatch.model.ListMetricsResult;
import com.amazonaws.services.cloudwatch.model.Metric;
```

## Code

```
final AmazonCloudWatch cw =
    AmazonCloudWatchClientBuilder.defaultClient();

ListMetricsRequest request = new ListMetricsRequest()
    .withMetricName(name)
    .withNamespace(namespace);

boolean done = false;

while(!done) {
    ListMetricsResult response = cw.listMetrics(request);

    for(Metric metric : response.getMetrics()) {
        System.out.printf(
            "Retrieved metric %s", metric.getMetricName());
    }

    request.setNextToken(response.getNextToken());

    if(response.getNextToken() == null) {
        done = true;
    }
}
```

Die Metriken werden in a zurückgegeben, indem die zugehörige Methode aufgerufen wird. [ListMetricsResult](#) `getMetrics` Eventuell werden die Ergebnisse seitenweise zurückgegeben. Um den nächsten Stapel Ergebnisse abzurufen, rufen Sie `setNextToken` beim Original-Anforderungsobjekt mit dem Rückgabewert der `getNextToken`-Methode des `ListMetricsResult`-Objekts auf. Übergeben Sie das geänderte Anforderungsobjekt dann an einen weiteren Aufruf von `listMetrics`.

## Weitere Informationen

- [ListMetrics](#) in der Amazon CloudWatch API-Referenz.

## Veröffentlichen benutzerdefinierter Metrikdaten

Eine Reihe von AWS Diensten veröffentlichen [ihre eigenen Metriken](#) in Namespaces, die mit "AWS" beginnen. Sie können benutzerdefinierte Metrikdaten auch mit Ihrem eigenen Namespace veröffentlichen (sofern dieser nicht mit "" beginnt). AWS

### Veröffentlichen benutzerdefinierter Metrikdaten

Um Ihre eigenen Metrikdaten zu veröffentlichen, rufen Sie die Methode `AmazonCloudWatchClient`'s `putMetricData` mit einem auf. [PutMetricDataRequest](#) Die `PutMetricDataRequest` muss den benutzerdefinierten Namespace enthalten, der für die Daten verwendet werden soll, und 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 Produkte reserviert. Amazon Web Services

### Importe

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatch;
import com.amazonaws.services.cloudwatch.AmazonCloudWatchClientBuilder;
import com.amazonaws.services.cloudwatch.model.Dimension;
import com.amazonaws.services.cloudwatch.model.MetricDatum;
import com.amazonaws.services.cloudwatch.model.PutMetricDataRequest;
import com.amazonaws.services.cloudwatch.model.PutMetricDataResult;
import com.amazonaws.services.cloudwatch.model.StandardUnit;
```

### Code

```
final AmazonCloudWatch cw =
    AmazonCloudWatchClientBuilder.defaultClient();

Dimension dimension = new Dimension()
    .withName("UNIQUE_PAGES")
    .withValue("URLS");

MetricDatum datum = new MetricDatum()
    .withMetricName("PAGES_VISITED")
    .withUnit(StandardUnit.None)
```

```
.withValue(data_point)
.withDimensions(dimension);

PutMetricDataRequest request = new PutMetricDataRequest()
    .withNamespace("SITE/TRAFFIC")
    .withMetricData(datum);

PutMetricDataResult response = cw.putMetricData(request);
```

## Weitere Informationen

- [Verwenden von Amazon CloudWatch Metriken](#) im Amazon CloudWatch Benutzerhandbuch.
- [AWS Namespaces](#) im Amazon CloudWatch Benutzerhandbuch.
- [PutMetricData](#) in der API-Referenz Amazon CloudWatch .

## Mit CloudWatch Alarmen arbeiten

### Einrichten eines Alarms

Um einen Alarm auf der Grundlage einer CloudWatch Metrik zu erstellen, rufen Sie die `AmazonCloudWatchClient.putMetricAlarm` Methode 'mit [PutMetricAlarmRequest](#) einer Angabe der Alarmbedingungen auf.

### Importe

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatch;
import com.amazonaws.services.cloudwatch.AmazonCloudWatchClientBuilder;
import com.amazonaws.services.cloudwatch.model.ComparisonOperator;
import com.amazonaws.services.cloudwatch.model.Dimension;
import com.amazonaws.services.cloudwatch.model.PutMetricAlarmRequest;
import com.amazonaws.services.cloudwatch.model.PutMetricAlarmResult;
import com.amazonaws.services.cloudwatch.model.StandardUnit;
import com.amazonaws.services.cloudwatch.model.Statistic;
```

### Code

```
final AmazonCloudWatch cw =
    AmazonCloudWatchClientBuilder.defaultClient();

Dimension dimension = new Dimension()
    .withName("InstanceId")
```

```
.withValue(instanceId);

PutMetricAlarmRequest request = new PutMetricAlarmRequest()
    .withAlarmName(alarmName)
    .withComparisonOperator(
        ComparisonOperator.GreaterThanThreshold)
    .withEvaluationPeriods(1)
    .withMetricName("CPUUtilization")
    .withNamespace("{AWS}/EC2")
    .withPeriod(60)
    .withStatistic(Statistic.Average)
    .withThreshold(70.0)
    .withActionsEnabled(false)
    .withAlarmDescription(
        "Alarm when server CPU utilization exceeds 70%")
    .withUnit(StandardUnit.Seconds)
    .withDimensions(dimension);

PutMetricAlarmResult response = cw.putMetricAlarm(request);
```

## Auflisten von Alarmen

Um die CloudWatch Alarme aufzulisten, die Sie erstellt haben, rufen Sie die `describeAlarms` Methode `AmazonCloudWatchClient`'s mit einer auf [DescribeAlarmsRequest](#), mit der Sie Optionen für das Ergebnis festlegen können.

### Importe

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatch;
import com.amazonaws.services.cloudwatch.AmazonCloudWatchClientBuilder;
import com.amazonaws.services.cloudwatch.model.DescribeAlarmsRequest;
import com.amazonaws.services.cloudwatch.model.DescribeAlarmsResult;
import com.amazonaws.services.cloudwatch.model.MetricAlarm;
```

### Code

```
final AmazonCloudWatch cw =
    AmazonCloudWatchClientBuilder.defaultClient();

boolean done = false;
DescribeAlarmsRequest request = new DescribeAlarmsRequest();

while(!done) {
```

```
DescribeAlarmsResult response = cw.describeAlarms(request);

for(MetricAlarm alarm : response.getMetricAlarms()) {
    System.out.printf("Retrieved alarm %s", alarm.getAlarmName());
}

request.setNextToken(response.getNextToken());

if(response.getNextToken() == null) {
    done = true;
}
}
```

Die Liste der Alarme kann abgerufen werden, indem Sie die `getMetricAlarms` Funktion aufrufen [DescribeAlarmsResult](#), die von zurückgegeben wird `describeAlarms`.

Eventuell werden die Ergebnisse seitenweise zurückgegeben. Um den nächsten Stapel Ergebnisse abzurufen, rufen Sie `setNextToken` beim Original-Anforderungsobjekt mit dem Rückgabewert der `getNextToken`-Methode des `DescribeAlarmsResult`-Objekts auf. Übergeben Sie das geänderte Anforderungsobjekt dann an einen weiteren Aufruf von `describeAlarms`.

#### Note

Sie können auch Alarme für eine bestimmte Metrik abrufen, indem Sie die `describeAlarmsForMetric` Methode `AmazonCloudWatchClient`'s verwenden. Sie lässt sich ähnlich wie `describeAlarms` nutzen.

## Löschen von Alarmen

Um CloudWatch Alarme zu löschen, rufen Sie die `AmazonCloudWatchClient` `deleteAlarms` Methode mit einer auf, [DeleteAlarmsRequest](#) die einen oder mehrere Namen von Alarmen enthält, die Sie löschen möchten.

### Importe

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatch;
import com.amazonaws.services.cloudwatch.AmazonCloudWatchClientBuilder;
import com.amazonaws.services.cloudwatch.model.DeleteAlarmsRequest;
import com.amazonaws.services.cloudwatch.model.DeleteAlarmsResult;
```

## Code

```
final AmazonCloudWatch cw =
    AmazonCloudWatchClientBuilder.defaultClient();

DeleteAlarmsRequest request = new DeleteAlarmsRequest()
    .withAlarmNames(alarm_name);

DeleteAlarmsResult response = cw.deleteAlarms(request);
```

## Weitere Informationen

- [Amazon CloudWatch Alarme erstellen](#) im Amazon CloudWatch Benutzerhandbuch
- [PutMetricAlarm](#) in der Amazon CloudWatch API-Referenz
- [DescribeAlarms](#) in der Amazon CloudWatch API-Referenz
- [DeleteAlarms](#) in der Amazon CloudWatch API-Referenz

## Verwenden von Alarmaktionen in CloudWatch

Mithilfe von CloudWatch Alarmaktionen können Sie Alarme erstellen, die Aktionen wie das automatische Stoppen, Beenden, Neustarten oder Wiederherstellen von Instanzen ausführen. Amazon EC2

### Note

Alarmaktionen können einem Alarm hinzugefügt werden, indem Sie bei der [PutMetricAlarmRequestErstellung eines](#) Alarms die `setAlarmActions` Methode verwenden.

## Aktivieren von Alarmaktionen

Um Alarmaktionen für einen CloudWatch Alarm zu aktivieren, rufen Sie den `AmazonCloudWatchClient enableAlarmActions` Befehl mit einem auf, der einen oder mehrere Namen von Alarmen [EnableAlarmActionsRequest](#) enthält, deren Aktionen Sie aktivieren möchten.

## Importe

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatch;
```

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatchClientBuilder;
import com.amazonaws.services.cloudwatch.model.EnableAlarmActionsRequest;
import com.amazonaws.services.cloudwatch.model.EnableAlarmActionsResult;
```

## Code

```
final AmazonCloudWatch cw =
    AmazonCloudWatchClientBuilder.defaultClient();

EnableAlarmActionsRequest request = new EnableAlarmActionsRequest()
    .withAlarmNames(alarm);

EnableAlarmActionsResult response = cw.enableAlarmActions(request);
```

## Deaktivieren von Alarmaktionen

Um Alarmaktionen für einen CloudWatch Alarm zu deaktivieren, rufen Sie die `AmazonCloudWatchClient`s `disableAlarmActions` mit einem auf, das einen oder mehrere Namen von Alarmen [DisableAlarmActionsRequest](#) enthält, deren Aktionen Sie deaktivieren möchten.

## Importe

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatch;
import com.amazonaws.services.cloudwatch.AmazonCloudWatchClientBuilder;
import com.amazonaws.services.cloudwatch.model.DisableAlarmActionsRequest;
import com.amazonaws.services.cloudwatch.model.DisableAlarmActionsResult;
```

## Code

```
final AmazonCloudWatch cw =
    AmazonCloudWatchClientBuilder.defaultClient();

DisableAlarmActionsRequest request = new DisableAlarmActionsRequest()
    .withAlarmNames(alarmName);

DisableAlarmActionsResult response = cw.disableAlarmActions(request);
```

## Weitere Informationen

- [Erstellen Sie im Amazon CloudWatch Benutzerhandbuch Alarme zum Stoppen, Beenden, Neustarten oder Wiederherstellen einer Instanz](#)

- [PutMetricAlarm](#) in der Amazon CloudWatch API-Referenz
- [EnableAlarmActions](#) in der Amazon CloudWatch API-Referenz
- [DisableAlarmActions](#) in der Amazon CloudWatch API-Referenz

## Ereignisse senden an CloudWatch

CloudWatch Events liefert nahezu in Echtzeit einen Stream von Systemereignissen, die Änderungen an AWS Ressourcen für Amazon EC2 Instanzen, Lambda Funktionen, Kinesis Streams, Amazon ECS Aufgaben, Schrittfunktionen Zustandsmaschinen, Amazon SNS Themen, Amazon SQS Warteschlangen oder integrierte Ziele beschreiben. Sie können Ereignisse zuordnen und sie zu einer oder mehreren Zielfunktionen oder Streams umleiten, indem Sie einfache Regeln nutzen.

### Hinzufügen von Ereignissen

Um benutzerdefinierte CloudWatch Ereignisse hinzuzufügen, rufen Sie die `AmazonCloudWatchEventsClient` `putEvents` 's-Methode mit einem [PutEventsRequest](#) Objekt auf, das ein oder mehrere [PutEventsRequestEntry](#) Objekte enthält, die Details zu jedem Ereignis bereitstellen. 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 com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEvents;
import com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsClientBuilder;
import com.amazonaws.services.cloudwatchevents.model.PutEventsRequest;
import com.amazonaws.services.cloudwatchevents.model.PutEventsRequestEntry;
import com.amazonaws.services.cloudwatchevents.model.PutEventsResult;
```

### Code

```
final AmazonCloudWatchEvents cwe =
    AmazonCloudWatchEventsClientBuilder.defaultClient();
```

```
final String EVENT_DETAILS =
    "{ \"key1\": \"value1\", \"key2\": \"value2\" }";

PutEventsRequestEntry request_entry = new PutEventsRequestEntry()
    .withDetail(EVENT_DETAILS)
    .withDetailType("sampleSubmitted")
    .withResources(resource_arn)
    .withSource("aws-sdk-java-cloudwatch-example");

PutEventsRequest request = new PutEventsRequest()
    .withEntries(request_entry);

PutEventsResult response = cwe.putEvents(request);
```

## Hinzufügen von Regeln

Um eine Regel zu erstellen oder zu aktualisieren, rufen Sie die `AmazonCloudWatchEventsClient` `putRule` Methode '[PutRuleRequest](#)' mit dem Namen der Regel und optionalen Parametern wie dem [Ereignismuster](#), der IAM Rolle, die der Regel zugeordnet werden soll, und einem [Planungsausdruck](#) auf, der beschreibt, wie oft die Regel ausgeführt wird.

### Importe

```
import com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEvents;
import com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsClientBuilder;
import com.amazonaws.services.cloudwatchevents.model.PutRuleRequest;
import com.amazonaws.services.cloudwatchevents.model.PutRuleResult;
import com.amazonaws.services.cloudwatchevents.model.RuleState;
```

### Code

```
final AmazonCloudWatchEvents cwe =
    AmazonCloudWatchEventsClientBuilder.defaultClient();

PutRuleRequest request = new PutRuleRequest()
    .withName(rule_name)
    .withRoleArn(role_arn)
    .withScheduleExpression("rate(5 minutes)")
    .withState(RuleState.ENABLED);

PutRuleResult response = cwe.putRule(request);
```

## Hinzufügen von Zielen

Ziele sind die Ressourcen, die beim Auslösen einer Regel aufgerufen werden. Zu den Beispielen gehören Amazon EC2 Instanzen, Lambda Funktionen, Kinesis Streams, Amazon ECS Aufgaben, Schrittfunktionen Zustandsmaschinen und integrierte Ziele.

Um einer Regel ein Ziel hinzuzufügen, rufen Sie die `AmazonCloudWatchEventsClient` `putTargets` 's-Methode mit einer auf, die die zu aktualisierende Regel und eine Liste von Zielen [PutTargetsRequest](#) enthält, die der Regel hinzugefügt werden sollen.

### Importe

```
import com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEvents;
import com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsClientBuilder;
import com.amazonaws.services.cloudwatchevents.model.PutTargetsRequest;
import com.amazonaws.services.cloudwatchevents.model.PutTargetsResult;
import com.amazonaws.services.cloudwatchevents.model.Target;
```

### Code

```
final AmazonCloudWatchEvents cwe =
    AmazonCloudWatchEventsClientBuilder.defaultClient();

Target target = new Target()
    .withArn(function_arn)
    .withId(target_id);

PutTargetsRequest request = new PutTargetsRequest()
    .withTargets(target)
    .withRule(rule_name);

PutTargetsResult response = cwe.putTargets(request);
```

## Weitere Informationen

- [Hinzufügen von Ereignissen mit PutEvents](#) im Amazon CloudWatch Events Benutzerhandbuch
- [Planen Sie Ausdrücke für Regeln](#) im Amazon CloudWatch Events Benutzerhandbuch
- [Ereignistypen für CloudWatch Ereignisse](#) im Amazon CloudWatch Events Benutzerhandbuch
- [Ereignisse und Ereignismuster](#) im Amazon CloudWatch Events Benutzerhandbuch
- [PutEvents](#) in der Amazon CloudWatch Events API-Referenz

- [PutTargets](#) in der Amazon CloudWatch Events API-Referenz
- [PutRule](#) in der Amazon CloudWatch Events API-Referenz

## DynamoDB Beispiele für die Verwendung der AWS SDK für Java

Dieser Abschnitt bietet Beispiele für die Programmierung von [DynamoDB](#) mithilfe des [AWS SDK für Java](#).

### Note

Die 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

- [Verwenden Sie AWS kontobasierte Endpunkte](#)
- [Arbeiten mit Tabellen in DynamoDB](#)
- [Arbeiten mit Elementen in DynamoDB](#)

## Verwenden Sie AWS kontobasierte Endpunkte

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

Um diese Funktion nutzen zu können, müssen Sie Version 1.12.771 oder höher von Version 1 verwenden. AWS SDK für Java Sie finden die neueste Version des SDK im zentralen [Maven-Repository](#). Sobald eine unterstützte Version des SDK aktiv ist, verwendet sie automatisch die neuen Endpunkte.

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

- Konfigurieren Sie einen DynamoDB-Dienstclient mit der `AccountIdEndpointMode` Einstellung auf `DISABLED`
- Legen Sie eine Umgebungsvariable fest.

- Legen Sie eine JVM-Systemeigenschaft fest.
- Aktualisieren Sie die Einstellung für die gemeinsam genutzte AWS Konfigurationsdatei.

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

```
ClientConfiguration config = new ClientConfiguration()
    .withAccountIdEndpointMode(AccountIdEndpointMode.DISABLED);
AWSCredentialsProvider credentialsProvider = new
    EnvironmentVariableCredentialsProvider();

AmazonDynamoDB dynamodb = AmazonDynamoDBClientBuilder.standard()
    .withClientConfiguration(config)
    .withCredentials(credentialsProvider)
    .withRegion(Regions.US_WEST_2)
    .build();
```

[Das AWS SDKs Referenzhandbuch zu Tools enthält weitere Informationen zu den letzten drei Konfigurationsoptionen.](#)

## Arbeiten mit Tabellen in DynamoDB

Tabellen sind die Container für alle Elemente in einer DynamoDB Datenbank. Bevor Sie Daten hinzufügen oder daraus entfernen können DynamoDB, müssen Sie eine Tabelle erstellen.

Für jede Tabelle definieren Sie:

- Einen Tabellennamen, der eindeutig für Ihr Konto und Ihre Region 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 nach 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.

- Werte zum bereitgestellten Durchsatz, die die Anzahl der reservierten Lese-Schreib-Kapazitätseinheiten für die Tabelle angeben.

#### Note

Amazon DynamoDB Die [Preisgestaltung](#) basiert auf den bereitgestellten Durchsatzwerten, die Sie für Ihre Tabellen festlegen. Reservieren Sie daher nur so viel Kapazität, wie Sie für Ihre Tabelle voraussichtlich benötigen.

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 `createTable` Methode des [DynamoDB Kunden](#), 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. Definieren Sie nur wichtige Tabellenattribute, wenn Sie Ihre DynamoDB Tabelle erstellen.

#### Note

Wenn eine Tabelle mit dem von Ihnen gewählten Namen bereits existiert, [AmazonServiceException](#) wird eine ausgelöst.

## Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.model.AttributeDefinition;
import com.amazonaws.services.dynamodbv2.model.CreateTableRequest;
import com.amazonaws.services.dynamodbv2.model.CreateTableResult;
import com.amazonaws.services.dynamodbv2.model.KeySchemaElement;
import com.amazonaws.services.dynamodbv2.model.KeyType;
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughput;
import com.amazonaws.services.dynamodbv2.model.ScalarAttributeType;
```

## Erstellen einer Tabelle mit einem einfachen Primärschlüssel

Dieser Code erstellt eine Tabelle mit einem einfachen Primärschlüssel ("Name").

### Code

```
CreateTableRequest request = new CreateTableRequest()
    .withAttributeDefinitions(new AttributeDefinition(
        "Name", ScalarAttributeType.S))
    .withKeySchema(new KeySchemaElement("Name", KeyType.HASH))
    .withProvisionedThroughput(new ProvisionedThroughput(
        new Long(10), new Long(10)))
    .withTableName(table_name);

final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

try {
    CreateTableResult result = ddb.createTable(request);
    System.out.println(result.getTableDescription().getTableName());
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Erstellen einer Tabelle mit einem zusammengesetzten Primärschlüssel

Füge ein weiteres hinzu [AttributeDefinition](#) und [KeySchemaElement](#) zu [CreateTableRequest](#).

### Code

```
CreateTableRequest request = new CreateTableRequest()
    .withAttributeDefinitions(
        new AttributeDefinition("Language", ScalarAttributeType.S),
        new AttributeDefinition("Greeting", ScalarAttributeType.S))
    .withKeySchema(
        new KeySchemaElement("Language", KeyType.HASH),
        new KeySchemaElement("Greeting", KeyType.RANGE))
    .withProvisionedThroughput(
        new ProvisionedThroughput(new Long(10), new Long(10)))
    .withTableName(table_name);
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Auflisten von Tabellen

Sie können die Tabellen in einer bestimmten Region auflisten, indem Sie die `listTables` Methode des [DynamoDB Clients](#) aufrufen.

### Note

Wenn die benannte Tabelle für Ihr Konto und Ihre Region nicht existiert, [ResourceNotFoundException](#) wird ausgelöst.

## Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.model.ListTablesRequest;
import com.amazonaws.services.dynamodbv2.model.ListTablesResult;
```

## Code

```
final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

ListTablesRequest request;

boolean more_tables = true;
String last_name = null;

while(more_tables) {
    try {
        if (last_name == null) {
            request = new ListTablesRequest().withLimit(10);
        }
        else {
            request = new ListTablesRequest()
                .withLimit(10)
                .withExclusiveStartTableName(last_name);
        }

        ListTablesResult table_list = ddb.listTables(request);
        List<String> table_names = table_list.getTableNames();
    }
}
```

```
if (table_names.size() > 0) {
    for (String cur_name : table_names) {
        System.out.format("* %s\n", cur_name);
    }
} else {
    System.out.println("No tables found!");
    System.exit(0);
}

last_name = table_list.getLastEvaluatedTableName();
if (last_name == null) {
    more_tables = false;
}
```

Standardmäßig werden bis zu 100 Tabellen pro Aufruf zurückgegeben. Verwenden Sie diese `getLastEvaluatedTableName` Option für das zurückgegebene [ListTablesResult](#) Objekt, um die zuletzt ausgewertete Tabelle abzurufen. Mit diesem Wert können Sie die Auflistung nach dem zuletzt zurückgegebenen Wert der vorherigen Auflistung beginnen.

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Beschreiben (Abrufen von Informationen zu) einer Tabelle

Rufen Sie die `describeTable` Methode des [DynamoDB Clients](#) auf.

### Note

Wenn die benannte Tabelle für Ihr Konto und Ihre Region nicht existiert, [ResourceNotFoundException](#) wird ausgelöst.

## Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.model.AttributeDefinition;
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughputDescription;
import com.amazonaws.services.dynamodbv2.model.TableDescription;
```

## Code

```
final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

try {
    TableDescription table_info =
        ddb.describeTable(table_name).getTable();

    if (table_info != null) {
        System.out.format("Table name   : %s\n",
            table_info.getTableName());
        System.out.format("Table ARN   : %s\n",
            table_info.getTableArn());
        System.out.format("Status      : %s\n",
            table_info.getTableStatus());
        System.out.format("Item count  : %d\n",
            table_info.getItemCount().longValue());
        System.out.format("Size (bytes): %d\n",
            table_info.getTableSizeBytes().longValue());

        ProvisionedThroughputDescription throughput_info =
            table_info.getProvisionedThroughput();
        System.out.println("Throughput");
        System.out.format("  Read Capacity : %d\n",
            throughput_info.getReadCapacityUnits().longValue());
        System.out.format("  Write Capacity: %d\n",
            throughput_info.getWriteCapacityUnits().longValue());

        List<AttributeDefinition> attributes =
            table_info.getAttributeDefinitions();
        System.out.println("Attributes");
        for (AttributeDefinition a : attributes) {
            System.out.format("  %s (%s)\n",
                a.getAttributeName(), a.getAttributeType());
        }
    }
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Ändern (Aktualisieren) einer Tabelle

Sie können die bereitgestellten Durchsatzwerte Ihrer Tabelle jederzeit ändern, indem Sie die `updateTable` Methode des [DynamoDB Clients](#) aufrufen.

### Note

Wenn die benannte Tabelle für Ihr Konto und Ihre Region nicht existiert, [ResourceNotFoundException](#) wird ausgelöst.

## Importe

```
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughput;
import com.amazonaws.AmazonServiceException;
```

## Code

```
ProvisionedThroughput table_throughput = new ProvisionedThroughput(
    read_capacity, write_capacity);

final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

try {
    ddb.updateTable(table_name, table_throughput);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Löschen einer Tabelle

Rufen Sie die `deleteTable` Methode des [DynamoDB Clients](#) auf und übergeben Sie ihr den Namen der Tabelle.

**Note**

Wenn die benannte Tabelle für Ihr Konto und Ihre Region nicht existiert, [ResourceNotFoundException](#) wird ausgelöst.

**Importe**

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
```

**Code**

```
final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

try {
    ddb.deleteTable(table_name);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

**Weitere Infos**

- [Richtlinien für die Arbeit mit Tabellen finden](#) Sie im Amazon DynamoDB Entwicklerhandbuch
- [Arbeiten mit Tabellen finden Sie DynamoDB im](#) Amazon DynamoDB Entwicklerhandbuch

**Arbeiten mit Elementen in DynamoDB**

DynamoDB In ist ein Element eine Sammlung von Attributen, von denen jedes einen Namen und einen Wert hat. Ein Attributwert kann eine Skalarfunktion, eine Gruppe oder ein Dokumenttyp sein. Weitere Informationen finden Sie unter [Benennungsregeln und Datentypen](#) im Amazon DynamoDB Entwicklerhandbuch.

## Abrufen (empfangen) eines Elements aus einer Tabelle

Rufen Sie die AmazonDynamo getItem DB-Methode auf und übergeben Sie ihr ein [GetItemRequest](#) Objekt mit dem Tabellennamen und dem Primärschlüsselwert des gewünschten Elements. Sie gibt ein [GetItemResult](#) Objekt zurück.

Sie können die getItem() Methode des zurückgegebenen GetItemResult Objekts verwenden, um eine [Map](#) von Schlüsselpaaren (String) und Wertpaaren ([AttributeValue](#)) abzurufen, die dem Element zugeordnet sind.

### Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.model.AttributeValue;
import com.amazonaws.services.dynamodbv2.model.GetItemRequest;
import java.util.HashMap;
import java.util.Map;
```

### Code

```
HashMap<String, AttributeValue> key_to_get =
    new HashMap<String, AttributeValue>();

key_to_get.put("DATABASE_NAME", new AttributeValue(name));

GetItemRequest request = null;
if (projection_expression != null) {
    request = new GetItemRequest()
        .withKey(key_to_get)
        .withTableName(table_name)
        .withProjectionExpression(projection_expression);
} else {
    request = new GetItemRequest()
        .withKey(key_to_get)
        .withTableName(table_name);
}

final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

try {
```

```
Map<String,AttributeValue> returned_item =
    ddb.getItem(request).getItem();
if (returned_item != null) {
    Set<String> keys = returned_item.keySet();
    for (String key : keys) {
        System.out.format("%s: %s\n",
            key, returned_item.get(key).toString());
    }
} else {
    System.out.format("No item found with the key %s!\n", name);
}
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Hinzufügen eines neuen Elements zu 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 existiert, [ResourceNotFoundException](#) wird ausgelöst.

## Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.model.AttributeValue;
import com.amazonaws.services.dynamodbv2.model.ResourceNotFoundException;
import java.util.ArrayList;
```

## Code

```
HashMap<String,AttributeValue> item_values =
    new HashMap<String,AttributeValue>();
```

```
item_values.put("Name", new AttributeValue(name));

for (String[] field : extra_fields) {
    item_values.put(field[0], new AttributeValue(field[1]));
}

final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

try {
    ddb.putItem(table_name, item_values);
} catch (ResourceNotFoundException e) {
    System.err.format("Error: The table \"%s\" can't be found.\n", table_name);
    System.err.println("Be sure that it exists and that you've typed its name correctly!");
    System.exit(1);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Aktualisieren eines vorhandenen Elements in einer Tabelle

Sie können ein Attribut für ein Element aktualisieren, das bereits in einer Tabelle vorhanden ist, indem Sie die `updateItem` Methode der AmazonDynamo Datenbank verwenden und dabei einen Tabellennamen, einen Primärschlüsselwert und eine Zuordnung der zu aktualisierenden Felder angeben.

### Note

Wenn die benannte Tabelle für Ihr Konto und Ihre Region nicht existiert oder wenn das Element, das durch den von Ihnen übergebenen Primärschlüssel identifiziert wurde, nicht existiert, [ResourceNotFoundException](#) wird ausgelöst.

## Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.model.AttributeAction;
```

```
import com.amazonaws.services.dynamodbv2.model.AttributeValue;
import com.amazonaws.services.dynamodbv2.model.AttributeValueUpdate;
import com.amazonaws.services.dynamodbv2.model.ResourceNotFoundException;
import java.util.ArrayList;
```

## Code

```
HashMap<String,AttributeValue> item_key =
    new HashMap<String,AttributeValue>();

item_key.put("Name", new AttributeValue(name));

HashMap<String,AttributeValueUpdate> updated_values =
    new HashMap<String,AttributeValueUpdate>();

for (String[] field : extra_fields) {
    updated_values.put(field[0], new AttributeValueUpdate(
        new AttributeValue(field[1]), AttributeAction.PUT));
}

final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

try {
    ddb.updateItem(table_name, item_key, updated_values);
} catch (ResourceNotFoundException e) {
    System.err.println(e.getMessage());
    System.exit(1);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Verwenden Sie die Dynamo-Klasse DBMapper

Die [AWS SDK für Java](#) stellt eine [DBMapperDynamo-Klasse](#) bereit, mit der Sie Ihre clientseitigen Klassen Tabellen zuordnen können. Amazon DynamoDB Um die [DBMapperDynamo-Klasse](#) zu verwenden, definieren Sie die Beziehung zwischen Elementen in einer DynamoDB Tabelle und ihren entsprechenden Objektinstanzen in Ihrem Code mithilfe von Anmerkungen (wie im folgenden Codebeispiel gezeigt). Mit der [DBMapperDynamo-Klasse](#) können Sie auf Ihre Tabellen zugreifen, verschiedene Erstellungs-, Lese-, Aktualisierungs- und Löschvorgänge (CRUD) ausführen und Abfragen ausführen.

**Note**

Mit der [DBMapperDynamo-Klasse](#) können Sie keine Tabellen erstellen, aktualisieren oder löschen.

**Importe**

```
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBAttribute;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBHashKey;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapper;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBTable;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBRangeKey;
import com.amazonaws.services.dynamodbv2.model.AmazonDynamoDBException;
```

**Code**

Das folgende Java-Codebeispiel zeigt Ihnen, wie Sie mithilfe der [DBMapperDynamo-Klasse](#) Inhalte zur Music-Tabelle hinzufügen. Nachdem der Inhalt der Tabelle hinzugefügt wurde, beachten Sie, dass ein Element mithilfe der Schlüssel Partition und Sortieren geladen wird. Anschließend wird das Element Auszeichnungen aktualisiert. Informationen zum Erstellen der Music-Tabelle finden Sie unter [Create a Table](#) im Amazon DynamoDB Developer Guide.

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
MusicItems items = new MusicItems();

try{
    // Add new content to the Music table
    items.setArtist(artist);
    items.setSongTitle(songTitle);
    items.setAlbumTitle(albumTitle);
    items.setAwards(Integer.parseInt(awards)); //convert to an int

    // Save the item
    DynamoDBMapper mapper = new DynamoDBMapper(client);
    mapper.save(items);

    // Load an item based on the Partition Key and Sort Key
    // Both values need to be passed to the mapper.load method
```

```
        String artistName = artist;
        String songQueryTitle = songTitle;

        // Retrieve the item
        MusicItems itemRetrieved = mapper.load(MusicItems.class, artistName,
songQueryTitle);
        System.out.println("Item retrieved:");
        System.out.println(itemRetrieved);

        // Modify the Award value
        itemRetrieved.setAwards(2);
        mapper.save(itemRetrieved);
        System.out.println("Item updated:");
        System.out.println(itemRetrieved);

        System.out.print("Done");
    } catch (AmazonDynamoDBException e) {
        e.printStackTrace();
    }
}

@DynamoDBTable(tableName="Music")
public static class MusicItems {

    //Set up Data Members that correspond to columns in the Music table
    private String artist;
    private String songTitle;
    private String albumTitle;
    private int awards;

    @DynamoDBHashKey(attributeName="Artist")
    public String getArtist() {
        return this.artist;
    }

    public void setArtist(String artist) {
        this.artist = artist;
    }

    @DynamoDBRangeKey(attributeName="SongTitle")
    public String getSongTitle() {
        return this.songTitle;
    }
}
```

```
public void setSongTitle(String title) {
    this.songTitle = title;
}

@DynamoDBAttribute(attributeName="AlbumTitle")
public String getAlbumTitle() {
    return this.albumTitle;
}

public void setAlbumTitle(String title) {
    this.albumTitle = title;
}

@DynamoDBAttribute(attributeName="Awards")
public int getAwards() {
    return this.awards;
}

public void setAwards(int awards) {
    this.awards = awards;
}
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Weitere Infos

- [Richtlinien für die Arbeit mit Elementen](#) im Amazon DynamoDB Entwicklerhandbuch
- [Arbeiten mit Elementen aus DynamoDB dem](#) Amazon DynamoDB Entwicklerhandbuch

## Amazon EC2 Beispiele für die Verwendung der AWS SDK für Java

Dieser Abschnitt enthält Beispiele für die Programmierung [Amazon EC2](#) mit dem AWS SDK für Java.

### Themen

- [Tutorial: Eine EC2 Instanz starten](#)
- [Verwenden von IAM-Rollen, um Zugriff auf AWS Ressourcen zu gewähren für Amazon EC2](#)
- [Tutorial: Amazon EC2 Spot-Instances](#)
- [Tutorial: Erweitertes Amazon EC2 Spot-Anforderungsmanagement](#)
- [Amazon EC2 Instanzen verwalten](#)

- [Verwendung von Elastic IP-Adressen in Amazon EC2](#)
- [Regionen und Verfügbarkeitszonen verwenden](#)
- [Mit Amazon EC2 Schlüsselpaaren arbeiten](#)
- [Arbeiten mit Sicherheitsgruppen in Amazon EC2](#)

## Tutorial: Eine EC2 Instanz starten

Dieses Tutorial zeigt, wie Sie mit AWS SDK für Java dem eine EC2 Instanz starten können.

### Themen

- [Voraussetzungen](#)
- [Eine Amazon EC2 Sicherheitsgruppe erstellen](#)
- [Erstellen eines Schlüsselpaares](#)
- [Eine Amazon EC2 Instanz ausführen](#)

## Voraussetzungen

Bevor Sie beginnen, stellen Sie sicher, dass Sie eine erstellt AWS-Konto und Ihre AWS Anmeldeinformationen eingerichtet haben. Weitere Informationen finden Sie unter [Erste Schritte mit](#) .

## Eine Amazon EC2 Sicherheitsgruppe erstellen

EC2-Classic geht in den Ruhestand

### Warning

Wir gehen in den Ruhestand EC2 -Classic am 15. August 2022. Wir empfehlen Ihnen, von EC2 -Classic zu einer VPC zu migrieren. Weitere Informationen finden Sie im Blogbeitrag [EC2-Classic-Classic-Classic Networking is Retiring](#) — So bereiten Sie sich vor.

Erstellen Sie eine Sicherheitsgruppe, die als virtuelle Firewall fungiert und den Netzwerkverkehr für eine oder mehrere Instanzen kontrolliert. EC2 Ordnet Ihre Amazon EC2 Instances standardmäßig einer Sicherheitsgruppe zu, die keinen eingehenden Datenverkehr zulässt. Sie können eine Sicherheitsgruppe erstellen, die es Ihren EC2 Instances ermöglicht, bestimmten Datenverkehr zu akzeptieren. Wenn Sie beispielsweise eine Verbindung zu einer Linux-Instance herstellen müssen,

muss die Sicherheitsgruppe so konfiguriert werden, dass SSH-Datenverkehr möglich ist. Sie können eine Sicherheitsgruppe mithilfe der Amazon EC2 Konsole oder der erstellen AWS SDK für Java.

Sie erstellen eine Sicherheitsgruppe für die Verwendung in EC2 -Classic oder VPC EC2. Weitere Informationen zu EC2 -Classic und EC2 VPC finden Sie unter [Unterstützte Plattformen](#) im Amazon EC2 Benutzerhandbuch für Linux-Instances.

Weitere Informationen zum Erstellen einer Sicherheitsgruppe mithilfe der Amazon EC2 Konsole finden Sie unter [Amazon EC2 Sicherheitsgruppen](#) im Amazon EC2 Benutzerhandbuch für Linux-Instances.

1. Erstellen und initialisieren Sie eine [CreateSecurityGroupRequest](#) Instanz. Verwenden Sie die [withGroupName](#) Methode, um den Namen der Sicherheitsgruppe festzulegen, und die Methode [withDescription](#), um die Beschreibung der Sicherheitsgruppe wie folgt festzulegen:

```
CreateSecurityGroupRequest csgr = new CreateSecurityGroupRequest();
csgr.withGroupName("JavaSecurityGroup").withDescription("My security group");
```

Der Name der Sicherheitsgruppe muss in der AWS Region, in der Sie Ihren Amazon EC2 Client initialisieren, eindeutig sein. Sie müssen US-ASCII-Zeichen für den Namen und die Beschreibung der Sicherheitsgruppe verwenden.

2. Übergeben Sie das Anforderungsobjekt als Parameter an die [createSecurityGroup](#) Methode. Die Methode gibt ein [CreateSecurityGroupResult](#) Objekt wie folgt zurück:

```
CreateSecurityGroupResult createSecurityGroupResult =
    amazonEC2Client.createSecurityGroup(csgr);
```

Wenn Sie versuchen, eine Sicherheitsgruppe mit dem gleichen Namen wie dem einer bereits vorhandenen Sicherheitsgruppe zu erstellen, gibt `createSecurityGroup` eine Ausnahme aus.

Standardmäßig lässt eine neue Sicherheitsgruppe keinen eingehenden Datenverkehr zu Ihrer Amazon EC2 Instance zu. Um eingehenden Datenverkehr zu erlauben, müssen Sie ausdrücklich eingehende Daten für die Sicherheitsgruppe autorisieren. Sie können eingehende Daten für einzelne IP-Adressen, für einen IP-Adressbereich, ein bestimmtes Protokoll sowie für TCP-/UDP-Ports autorisieren.

1. Erstellen und initialisieren Sie eine Instanz [IpPermission](#). Verwenden Sie die Methode [withIPv4Ranges](#), um den Bereich der IP-Adressen festzulegen, für den der Zugriff autorisiert

werden soll, und verwenden Sie die Methode, um das [withIpProtocol](#)IP-Protokoll festzulegen. Verwenden Sie die [withToPort](#)Methoden [withFromPort](#)und, um den Portbereich anzugeben, für den der eingehende Datenverkehr autorisiert werden soll, wie folgt:

```
IpPermission ipPermission =
    new IpPermission();

IpRange ipRange1 = new IpRange().withCidrIp("111.111.111.111/32");
IpRange ipRange2 = new IpRange().withCidrIp("150.150.150.150/32");

ipPermission.withIpv4Ranges(Arrays.asList(new IpRange[] {ipRange1, ipRange2}))
    .withIpProtocol("tcp")
    .withFromPort(22)
    .withToPort(22);
```

Alle im `IpPermission`-Objekt angegebenen Bedingungen müssen erfüllt sein, damit eingehende Daten zugelassen werden.

Geben Sie die IP-Adresse über CIDR-Notation an. Wenn Sie das Protokoll als TCP/UDP angeben, müssen Sie einen Quell- und Ziel-Port festlegen. Sie können Ports nur bei Angabe von TCP oder UDP autorisieren.

- Erstellen und initialisieren Sie eine Instanz. [AuthorizeSecurityGroupIngressRequest](#) Verwenden Sie die `withGroupName` Methode, um den Namen der Sicherheitsgruppe anzugeben, und übergeben Sie das `IpPermission` Objekt, das Sie zuvor initialisiert haben, wie folgt an die [withIpPermissions](#)Methode:

```
AuthorizeSecurityGroupIngressRequest authorizeSecurityGroupIngressRequest =
    new AuthorizeSecurityGroupIngressRequest();

authorizeSecurityGroupIngressRequest.withGroupName("JavaSecurityGroup")
    .withIpPermissions(ipPermission);
```

- Übergeben Sie das Anforderungsobjekt wie folgt an die [authorizeSecurityGroupIngress-Methode](#):

```
amazonEC2Client.authorizeSecurityGroupIngress(authorizeSecurityGroupIngressRequest);
```

Wenn Sie `authorizeSecurityGroupIngress` mit IP-Adressen aufrufen, für die eingehende Daten bereits autorisiert sind, gibt diese Methode eine Ausnahme aus. Erstellen und initialisieren

Sie vor dem Aufruf ein neues `IpPermission` Objekt, um den Zugriff für verschiedene Ports und IPs Protokolle zu autorisieren. `AuthorizeSecurityGroupIngress`

Immer wenn Sie die Methoden [authorizeSecurityGroupIngress](#) oder [authorizeSecurityGroupEgress](#) aufrufen, wird Ihrer Sicherheitsgruppe eine Regel hinzugefügt.

## Erstellen eines Schlüsselpaares

Sie müssen ein key pair angeben, wenn Sie eine EC2 Instance starten, und dann den privaten Schlüssel des key pair angeben, wenn Sie eine Verbindung mit der Instance herstellen. Sie können ein Schlüsselpaar erstellen oder ein vorhandenes Schlüsselpaar verwenden, das Sie beim Start anderer Instances genutzt haben. Weitere Informationen finden Sie unter [Amazon EC2 Schlüsselpaare](#) im Amazon EC2 Benutzerhandbuch für Linux-Instances.

1. Erstellen und initialisieren Sie eine [CreateKeyPairRequest](#) Instanz. Verwenden Sie die [withKeyName](#) Methode, um den Namen des key pair wie folgt festzulegen:

```
CreateKeyPairRequest createKeyPairRequest = new CreateKeyPairRequest();
createKeyPairRequest.withKeyName(keyName);
```

### Important

Namen von Schlüsselpaaren müssen eindeutig sein. Wenn Sie versuchen, ein Schlüsselpaar mit dem gleichen Namen wie dem eines bereits vorhandenen Schlüsselpaares zu erstellen, wird eine Ausnahme ausgelöst.

2. Übergeben Sie das Anforderungsobjekt an die [createKeyPair](#) Methode. Die Methode gibt wie folgt eine [CreateKeyPairResult](#) Instanz zurück:

```
CreateKeyPairResult createKeyPairResult =
amazonEC2Client.createKeyPair(createKeyPairRequest);
```

3. Rufen Sie die [getKeyPair](#) Methode des Ergebnisobjekts auf, um ein [KeyPair](#) Objekt zu erhalten. Rufen Sie die [getKeyMaterial](#) Methode des `KeyPair` Objekts wie folgt auf, um den unverschlüsselten PEM-codierten privaten Schlüssel abzurufen:

```
KeyPair keyPair = new KeyPair();
```

```
keyPair = createKeyPairResult.getKeyPair();  
  
String privateKey = keyPair.getKeyMaterial();
```

## Eine Amazon EC2 Instanz ausführen

Gehen Sie wie folgt vor, um eine oder mehrere identisch konfigurierte EC2 Instances von demselben Amazon Machine Image (AMI) aus zu starten. Nachdem Sie Ihre EC2 Instances erstellt haben, können Sie deren Status überprüfen. Nachdem Ihre EC2 Instances ausgeführt wurden, können Sie eine Verbindung zu ihnen herstellen.

1. Erstellen und initialisieren Sie eine [RunInstancesRequest](#) Instanz. Stellen Sie sicher, dass das AMI, das Schlüsselpaar und die Sicherheitsgruppe, die Sie angeben, in der beim Erstellen des Client-Objekts angegebenen Region vorhanden sind.

```
RunInstancesRequest runInstancesRequest =  
    new RunInstancesRequest();  
  
runInstancesRequest.withImageId("ami-a9d09ed1")  
                    .withInstanceType(InstanceType.T1Micro)  
                    .withMinCount(1)  
                    .withMaxCount(1)  
                    .withKeyName("my-key-pair")  
                    .withSecurityGroups("my-security-group");
```

### [withImageId](#)

- Die ID des AMI. Unter Amazon [Machine Image \(AMI\) erfahren Sie, wie Sie die von Amazon AMIs bereitgestellten Inhalte](#) finden oder Ihre eigenen erstellen können.

### [withInstanceType](#)

- Ein Instance-Typ, der kompatibel mit dem angegebenen AMI ist. Weitere Informationen finden Sie unter [Instance-Typen](#) im Amazon EC2 Benutzerhandbuch für Linux-Instances.

### [withMinCount](#)

- Die Mindestanzahl der zu startenden EC2 Instances. Wenn es sich dabei um mehr Instances handelt, als in der Ziel-Availability Zone gestartet werden Amazon EC2 können, werden keine Instances Amazon EC2 gestartet.

### [withMaxCount](#)

- Die maximale Anzahl der zu startenden EC2 Instances. Wenn es sich dabei um mehr Instances handelt, als in der Ziel-Availability Zone Amazon EC2 gestartet werden können, wird die größtmögliche Anzahl von oben genannten Instances gestartet `MinCount`. Sie können zwischen 1 und der maximalen Anzahl der Instances starten, die für Sie beim jeweiligen Instance-Typ zulässig sind. Weitere Informationen finden Sie unter [Wie viele Instances kann ich ausführen? Amazon EC2](#) in den Amazon EC2 allgemeinen FAQs.

### [withKeyName](#)

- Der Name des EC2 key pair. Wenn Sie eine Instance ohne Angabe eines Schlüsselpaars starten, können Sie sich nicht mit ihr verbinden. Weitere Informationen finden Sie unter [Erstellen eines Schlüsselpaars](#).

### [withSecurityGroups](#)

- Eine oder mehrere Sicherheitsgruppen. Weitere Informationen finden Sie unter [Amazon EC2 Sicherheitsgruppe erstellen](#).

2. Starten Sie die Instances, indem Sie das Anforderungsobjekt an die [runInstances](#)-Methode übergeben. Die Methode gibt ein [RunInstancesResult](#) Objekt wie folgt zurück:

```
RunInstancesResult result = amazonEC2Client.runInstances(  
    runInstancesRequest);
```

Sobald die Instance ausgeführt wird, können Sie sich mit dem Schlüsselpaar mit ihr verbinden. Weitere Informationen finden Sie unter [Connect to Your Linux Instance](#) im Amazon EC2 Benutzerhandbuch für Linux-Instances.

## Verwenden von IAM-Rollen, um Zugriff auf AWS Ressourcen zu gewähren für Amazon EC2

Alle Anfragen an Amazon Web Services (AWS) müssen kryptografisch signiert werden, wobei die Anmeldeinformationen verwendet werden müssen, die von ausgestellt wurden. AWS Sie können IAM-Rollen verwenden, um bequem sicheren Zugriff auf AWS Ressourcen von Ihren Instances aus zu gewähren. Amazon EC2

Dieses Thema enthält Informationen zur Verwendung von IAM-Rollen mit Java SDK-Anwendungen, die auf ausgeführt werden. Amazon EC2 Weitere Informationen zu IAM-Instanzen finden Sie unter [IAM-Rollen für Amazon EC2 im Amazon EC2 Benutzerhandbuch für Linux-Instances](#).

## Die Standardanbieterketten und Instanzprofile EC2

Wenn Ihre Anwendung mithilfe des Standardkonstruktors einen AWS Client erstellt, sucht der Client mithilfe der standardmäßigen Anbieterkette für Anmeldeinformationen in der folgenden Reihenfolge nach Anmeldeinformationen:

1. in den Java-Systemeigenschaften: `aws.accessKeyId` und `aws.secretKey`
2. in System-Umgebungsvariablen: `AWS_ACCESS_KEY_ID` und `AWS_SECRET_ACCESS_KEY`
3. in der Standarddatei für Anmeldeinformationen (der Speicherort dieser Datei hängt von der jeweiligen Plattform ab)
4. Anmeldeinformationen, die über den Amazon EC2 Container-Service bereitgestellt werden, wenn die `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` Umgebungsvariable gesetzt ist und der Sicherheitsmanager über die Zugriffsberechtigung für die Variable verfügt.
5. In den Instanzprofil-Anmeldeinformationen, die in den Instanz-Metadaten enthalten sind, die der IAM-Rolle für die EC2 Instanz zugeordnet sind.
6. Web-Identitätstoken-Anmeldeinformationen aus der Umgebung oder dem Container.

Der Schritt mit den Anmeldeinformationen für das Instanzprofil in der Standardanbieterkette ist nur verfügbar, wenn Sie Ihre Anwendung auf einer Amazon EC2 Instance ausführen, bietet jedoch die größte Benutzerfreundlichkeit und die beste Sicherheit bei der Arbeit mit Amazon EC2 Instances. Sie können eine [InstanceProfileCredentialsProvider](#) Instanz auch direkt an den Client-Konstruktor übergeben, um die Anmeldeinformationen für das Instanzprofil abzurufen, ohne die gesamte Standardanbieterkette durchlaufen zu müssen.

Zum Beispiel:

```
AmazonS3 s3 = AmazonS3ClientBuilder.standard()
    .withCredentials(new InstanceProfileCredentialsProvider(false))
    .build();
```

Bei diesem Ansatz ruft das SDK temporäre AWS Anmeldeinformationen ab, die dieselben Berechtigungen haben wie die, die der IAM-Rolle zugeordnet sind, die der Amazon EC2 Instanz in ihrem Instanzprofil zugeordnet ist. Diese Anmeldeinformationen sind zwar temporär und würden

irgendwann ablaufen, aktualisiert sie jedoch `InstanceProfileCredentialsProvider` regelmäßig für Sie, sodass die abgerufenen Anmeldeinformationen weiterhin Zugriff auf ermöglichen.  
AWS

### Important

Die Anmeldeinformationen werden nur dann automatisch aktualisiert, wenn Sie den standardmäßigen Client-Konstruktor verwenden, der seinen eigenen `InstanceProfileCredentialsProvider` als Teil der standardmäßigen Anbieterkette erstellt, oder wenn Sie eine `InstanceProfileCredentialsProvider`-Instance direkt an den Client-Konstruktor übergeben. Wenn Sie Anmeldeinformationen des Instance-Profils auf andere Weise abrufen oder übergeben, sind Sie selbst für die Überprüfung und ggf. für die Aktualisierung abgelaufener Anmeldeinformationen zuständig.

Wenn der Client-Konstruktor mithilfe der Anmeldeinformationsanbieterkette keine Anmeldeinformationen finden kann, gibt er eine aus. [AmazonClientException](#)

## Exemplarische Vorgehensweise: Verwenden von IAM-Rollen für Instanzen EC2

In der folgenden exemplarischen Vorgehensweise wird gezeigt, wie Sie Amazon S3 mithilfe einer IAM-Rolle ein Objekt abrufen können, um den Zugriff zu verwalten.

### Erstellen einer IAM-Rolle

Erstellen Sie eine IAM-Rolle, die nur Lesezugriff auf gewährt. Amazon S3

1. Öffnen Sie die [IAM-Konsole](#).
2. Wechseln Sie im Navigationsbereich zu Roles (Rollen) und klicken Sie auf Create New Role (Neue Rolle erstellen).
3. Geben Sie einen Namen für die Rolle ein und klicken Sie dann auf Next Step. Merken Sie sich diesen Namen, da Sie ihn benötigen, wenn Sie Ihre Instance starten. Amazon EC2
4. Wählen Sie auf der Seite „Rollentyp auswählen“ unter AWS-Service Rollen die Option aus Amazon EC2 .
5. Wählen Sie auf der Seite „Berechtigungen festlegen“ unter Richtlinienvorlage auswählen die Option Amazon S3 Schreibgeschützter Zugriff und dann Nächster Schritt aus.
6. Wählen Sie auf der Seite Review (Prüfen) Create Role (Rolle erstellen) aus.

## Starten Sie eine EC2 Instance und geben Sie Ihre IAM-Rolle an

Sie können eine Amazon EC2 Instance mit einer IAM-Rolle über die Amazon EC2 Konsole oder die AWS CLI starten. AWS SDK für Java

- Um eine Amazon EC2 Instance über die Konsole zu starten, folgen Sie den Anweisungen unter [Erste Schritte mit Amazon EC2 Linux-Instances](#) im Amazon EC2 Benutzerhandbuch für Linux-Instances.

Wenn Sie die Seite Review Instance Launch erreichen, klicken Sie auf Edit instance details. Wählen Sie unter IAM-Rolle die IAM-Rolle aus, die Sie zuvor erstellt haben. Befolgen Sie die Anweisungen und schließen Sie den Vorgang ab.

### Note

Zum Herstellen einer Verbindung mit der Instance müssen Sie eine Sicherheitsgruppe und ein Schlüsselpaar neu erstellen oder vorhandene Anmeldeinformationen auswählen.

- Informationen zum Starten einer Amazon EC2 Instance mit einer IAM-Rolle mithilfe von `awscli` finden Sie unter Instanz [ausführen](#). AWS SDK für Java Amazon EC2

## Erstellen Ihrer Anwendung

Lassen Sie uns die Beispielanwendung erstellen, die auf der EC2 Instance ausgeführt werden soll. Zuerst erstellen Sie ein Verzeichnis, in dem Sie die Tutorial-Dateien abspeichern können (z. B. `GetS3ObjectApp`).

Kopieren Sie als Nächstes die AWS SDK für Java Bibliotheken in Ihr neu erstelltes Verzeichnis. Wenn Sie die AWS SDK für Java in Ihr `~/Downloads` Verzeichnis heruntergeladen haben, können Sie sie mit den folgenden Befehlen kopieren:

```
cp -r ~/Downloads/aws-java-sdk-{1.7.5}/lib .
cp -r ~/Downloads/aws-java-sdk-{1.7.5}/third-party .
```

Legen Sie eine neue Datei namens `GetS3Object.java` an und fügen Sie den folgenden Code ein:

```
import java.io.*;

import com.amazonaws.auth.*;
import com.amazonaws.services.s3.*;
```

```
import com.amazonaws.services.s3.model.*;
import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;

public class GetS3Object {
    private static final String bucketName = "text-content";
    private static final String key = "text-object.txt";

    public static void main(String[] args) throws IOException
    {
        AmazonS3 s3Client = AmazonS3ClientBuilder.defaultClient();

        try {
            System.out.println("Downloading an object");
            S3Object s3object = s3Client.getObject(
                new GetObjectRequest(bucketName, key));
            displayTextInputStream(s3object.getObjectContent());
        }
        catch(AmazonServiceException ase) {
            System.err.println("Exception was thrown by the service");
        }
        catch(AmazonClientException ace) {
            System.err.println("Exception was thrown by the client");
        }
    }

    private static void displayTextInputStream(InputStream input) throws IOException
    {
        // Read one text line at a time and display.
        BufferedReader reader = new BufferedReader(new InputStreamReader(input));
        while(true)
        {
            String line = reader.readLine();
            if(line == null) break;
            System.out.println( "    " + line );
        }
        System.out.println();
    }
}
```

Legen Sie eine weitere neue Datei namens `build.xml` an und fügen Sie folgende Zeilen ein:

```
<project name="Get {S3} Object" default="run" basedir=".">
```

```
<path id="aws.java.sdk.classpath">
  <fileset dir="./lib" includes="**/*.jar"/>
  <fileset dir="./third-party" includes="**/*.jar"/>
  <pathelement location="lib"/>
  <pathelement location="."/>
</path>

<target name="build">
<javac debug="true"
  includeantruntime="false"
  srcdir="."
  destdir="."
  classpathref="aws.java.sdk.classpath"/>
</target>

<target name="run" depends="build">
  <java classname="GetS3Object" classpathref="aws.java.sdk.classpath" fork="true"/>
</target>
</project>
```

Erstellen und starten Sie das geänderte Programm. Beachten Sie, dass keine Anmeldeinformationen im Programm gespeichert werden. Daher wird der Code ausgelöst, sofern Sie Ihre AWS Anmeldeinformationen nicht bereits angegeben haben `AmazonServiceException`. Zum Beispiel:

```
$ ant
Buildfile: /path/to/my/GetS3ObjectApp/build.xml

build:
  [javac] Compiling 1 source file to /path/to/my/GetS3ObjectApp

run:
  [java] Downloading an object
  [java] AmazonServiceException

BUILD SUCCESSFUL
```

Übertragen Sie das kompilierte Programm auf Ihre EC2 Instance

Übertragen Sie das Programm zusammen mit den AWS SDK für Java Bibliotheken mithilfe von `Secure Copy ()` auf Ihre Amazon EC2 Instanz. Die Reihenfolge der Befehle sieht in etwa wie folgt aus:

```
scp -p -i {my-key-pair}.pem GetS3Object.class ec2-user@{public_dns}:GetS3Object.class
scp -p -i {my-key-pair}.pem build.xml ec2-user@{public_dns}:build.xml
scp -r -p -i {my-key-pair}.pem lib ec2-user@{public_dns}:lib
scp -r -p -i {my-key-pair}.pem third-party ec2-user@{public_dns}:third-party
```

### Note

Abhängig von der verwendeten Linux-Distribution lautet der user name (Benutzername) "ec2-user", "root" oder "ubuntu". Um den öffentlichen DNS-Namen Ihrer Instance abzurufen, öffnen Sie die [EC2 Konsole](#) und suchen Sie auf der Registerkarte Beschreibung nach dem Wert Public DNS (z. B. `ec2-198-51-100-1.compute-1.amazonaws.com`).

Bei den vorhergehenden Befehlen:

- ist `GetS3Object.class` Ihr kompiliertes Programm,
- `build.xml` ist die Ant-Datei zum Erstellen und Ausführen Ihres Programms und
- die Verzeichnisse `lib` und `third-party` sind die entsprechenden Bibliotheksordner aus dem AWS SDK für Java.
- Der `-r` Schalter gibt an, dass eine rekursive Kopie des gesamten Inhalts der `third-party` Verzeichnisse `library` und in der AWS SDK für Java Distribution erstellt werden `scp` soll.
- Der Schalter `-p` sorgt dafür, dass `scp` die Berechtigungen der Quelldateien beibehalten soll, während diese zum Ziel kopiert werden.

### Note

Der `-p` Switch funktioniert nur unter Linux, macOS oder Unix. Wenn Sie Dateien von Windows kopieren, müssen Sie die Dateiberechtigungen auf Ihrer Instance mit dem folgenden Befehl korrigieren:

```
chmod -R u+rxw GetS3Object.class build.xml lib third-party
```

Führen Sie das Beispielprogramm auf der EC2 Instanz aus

Um das Programm auszuführen, stellen Sie eine Verbindung zu Ihrer Amazon EC2 Instance her. Weitere Informationen finden Sie unter [Connect to Your Linux Instance](#) im Amazon EC2 Benutzerhandbuch für Linux-Instances.

Wenn **ant** auf Ihrer Instance nicht verfügbar ist, installieren Sie es mit folgendem Befehl:

```
sudo yum install ant
```

Führen Sie das Programm dann mithilfe von ant wie folgt aus:

```
ant run
```

Das Programm schreibt den Inhalt Ihres Amazon S3 Objekts in Ihr Befehlsfenster.

## Tutorial: Amazon EC2 Spot-Instances

### Übersicht

Spot-Instances ermöglichen es Ihnen, auf ungenutzte Amazon Elastic Compute Cloud (Amazon EC2) Kapazität bis zu 90% gegenüber dem Preis für On-Demand-Instances zu bieten und die erworbenen Instances so lange laufen zu lassen, wie Ihr Gebot den aktuellen Spot-Preis übersteigt. Amazon EC2 ändert den Spot-Preis in regelmäßigen Abständen auf der Grundlage von Angebot und Nachfrage, und Kunden, deren Gebote dem Preis entsprechen oder ihn übertreffen, erhalten Zugriff auf die verfügbaren Spot-Instances. Wie mit On-Demand-Instances und Reserved Instances erhalten Sie mit Spot-Instances eine weitere Möglichkeit, mehr Rechenkapazität zu erhalten.

Spot-Instances können Ihre Amazon EC2 Kosten für Stapelverarbeitung, wissenschaftliche Forschung, Bildverarbeitung, Videokodierung, Daten- und Webcrawling, Finanzanalysen und Tests erheblich senken. Darüber hinaus ermöglichen Spot-Instances Zugriff auf große Mengen an zusätzlicher Kapazität in Situationen, in denen diese Kapazität nicht dringend benötigt wird.

Senden Sie zur Nutzung von Spot-Instances eine Spot-Instance-Anforderung und geben Sie den Höchstpreis an, den Sie pro Instance-Stunde zu zahlen bereit sind; dies ist Ihr Gebot. Wenn Ihr Höchstgebot den aktuellen Spot-Preis übersteigt, wird Ihrer Anforderung stattgegeben. Ihre Instances werden so lange ausgeführt, bis Sie sie entweder beenden oder bis der Spot-Preis Ihr Höchstgebot übersteigt.

## Wichtiger Hinweis:

- Sie zahlen oft weniger pro Stunde als Sie geboten haben. Amazon EC2 passt den Spotpreis regelmäßig an, wenn Anfragen eingehen und sich das verfügbare Angebot ändert. Alle Benutzer bezahlen für diesen Zeitraum denselben Spot-Preis, unabhängig davon, ob ihr Gebot höher lag. Daher zahlen Sie möglicherweise weniger als Ihr Gebot, aber Sie zahlen nie mehr als das Gebot.
- Wenn Sie Spot-Instances ausführen und Ihr Gebot nicht mehr mindestens dem aktuellen Spot-Preis entspricht, werden Ihre Instances beendet. Daher sollten Sie sicherstellen, dass Ihre Workloads und Anwendungen flexibel genug sind, um aus dieser gelegentlichen Kapazität Nutzen zu ziehen.

Spot-Instances verhalten sich während der Ausführung genau wie andere Amazon EC2 Instances, und wie andere Amazon EC2 Instances können Spot-Instances beendet werden, wenn Sie sie nicht mehr benötigen. Beim Beenden einer Instance zahlen Sie für die angefangene Stunde (wie bei On-Demand- und Reserved Instances). Wenn der Spot-Preis jedoch über Ihrem Gebot liegt und Ihre Instance bis gekündigt wird Amazon EC2, wird Ihnen keine Teilstunde der Nutzung in Rechnung gestellt.

In diesem Tutorial wird gezeigt, wie AWS SDK für Java Sie Folgendes tun können.

- Senden einer Spot-Anfrage
- Ermitteln, wann die Spot-Anfrage erfüllt wird
- Abbrechen der Spot-Anfrage
- Beenden von dazugehörigen Instances

## Voraussetzungen

Um dieses Tutorial verwenden zu können, müssen Sie das AWS SDK für Java installiert haben und die grundlegenden Installationsvoraussetzungen erfüllen. Weitere Informationen finden [Sie unter Einrichten](#) von AWS SDK für Java

## Schritt 1: Einrichten Ihrer Anmeldeinformationen

Um mit der Verwendung dieses Codebeispiels zu beginnen, müssen Sie AWS Anmeldeinformationen einrichten. Anweisungen dazu finden Sie unter [AWS Zugangsdaten und Region für die Entwicklung einrichten](#).

**Note**

Wir empfehlen, dass Sie die Anmeldeinformationen eines IAM-Benutzers für diese Werte nutzen. Weitere Informationen finden Sie unter [Registrierung für einen IAM-Benutzer AWS und Erstellen eines IAM-Benutzers](#).

Nachdem Sie Ihre Einstellungen eingerichtet haben, können Sie mit dem Beispiel-Code loslegen.

## Schritt 2: Einrichten einer Sicherheitsgruppe

Eine Sicherheitsgruppe agiert als Firewall, die den zulässigen Verkehr zu und von einer Gruppe Instances steuert. Standardmäßig wird eine Instance ohne eine Sicherheitsgruppe gestartet. Sämtlicher eingehender IP-Datenverkehr auf allen TCP-Ports wird daher verweigert. Vor dem Absenden unserer Spot-Anforderung richten wir also eine Sicherheitsgruppe ein, die den nötigen Netzwerkverkehr zulässt. Für die Zwecke dieses Tutorials erstellen wir eine neue Sicherheitsgruppe mit dem Namen "GettingStarted", die Secure Shell (SSH) -Verkehr von der IP-Adresse aus ermöglicht, von der aus Sie Ihre Anwendung ausführen. Zur Einrichtung einer neuen Sicherheitsgruppe sollten Sie das folgende Codebeispiel einschließen oder ausführen. Dadurch wird die Sicherheitsgruppe per Programm eingerichtet.

Nachdem wir ein AmazonEC2 Client-Objekt erstellt haben, erstellen wir ein `CreateSecurityGroupRequest` Objekt mit dem Namen "GettingStarted" und einer Beschreibung für die Sicherheitsgruppe. Anschließend wird die `ec2.createSecurityGroup`-API zum Erstellen der Gruppe aufgerufen.

Zum Aktivieren des Zugriffs auf die Gruppe erstellen wir ein `ipPermission`-Objekt, bei dem der IP-Adressbereich des Subnetzes auf die CIDR-Darstellung des Subnetzes des lokalen Computers festgelegt ist. Das Suffix `/10` bei der IP-Adresse zeigt das Subnetz für die angegebene IP-Adresse an. Anschließend konfigurieren wir auch das `ipPermission`-Objekt mit dem TCP-Protokoll und dem Port 22 (SSH). Im letzten Schritt wird `ec2.authorizeSecurityGroupIngress` mit dem Namen der Sicherheitsgruppe und dem `ipPermission`-Objekt aufgerufen.

```
// Create the AmazonEC2 client so we can call various APIs.
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

// Create a new security group.
try {
    CreateSecurityGroupRequest securityGroupRequest = new
    CreateSecurityGroupRequest("GettingStartedGroup", "Getting Started Security Group");
```

```
    ec2.createSecurityGroup(securityGroupRequest);
} catch (AmazonServiceException ase) {
    // Likely this means that the group is already created, so ignore.
    System.out.println(ase.getMessage());
}

String ipAddr = "0.0.0.0/0";

// Get the IP of the current host, so that we can limit the Security
// Group by default to the ip range associated with your subnet.
try {
    InetAddress addr = InetAddress.getLocalHost();

    // Get IP Address
    ipAddr = addr.getHostAddress()+"/10";
} catch (UnknownHostException e) {
}

// Create a range that you would like to populate.
ArrayList<String> ipRanges = new ArrayList<String>();
ipRanges.add(ipAddr);

// Open up port 22 for TCP traffic to the associated IP
// from above (e.g. ssh traffic).
ArrayList<IpPermission> ipPermissions = new ArrayList<IpPermission> ();
IpPermission ipPermission = new IpPermission();
ipPermission.setIpProtocol("tcp");
ipPermission.setFromPort(new Integer(22));
ipPermission.setToPort(new Integer(22));
ipPermission.setIpRanges(ipRanges);
ipPermissions.add(ipPermission);

try {
    // Authorize the ports to the used.
    AuthorizeSecurityGroupIngressRequest ingressRequest =
        new AuthorizeSecurityGroupIngressRequest("GettingStartedGroup",ipPermissions);
    ec2.authorizeSecurityGroupIngress(ingressRequest);
} catch (AmazonServiceException ase) {
    // Ignore because this likely means the zone has
    // already been authorized.
    System.out.println(ase.getMessage());
}
```

Hinweis: Sie müssen diese Anwendung nur einmal ausführen, um eine neue Sicherheitsgruppe zu erstellen.

Sie können die Sicherheitsgruppe auch mithilfe von AWS Toolkit for Eclipse erstellen. Weitere Informationen finden Sie unter [Sicherheitsgruppen verwalten von AWS Cost Explorer](#).

### Schritt 3: Senden Ihrer Spot-Instance-Anforderung

Zum Senden einer Spot-Anforderung sollten Sie zuerst den Instance-Typ, das Amazon Machine Image (AMI) und den Höchstpreis für Ihr Gebot festlegen. Sie müssen auch die zuvor konfigurierte Sicherheitsgruppe aufnehmen, damit Sie sich bei Bedarf bei der Instance anmelden können.

Es stehen mehrere Instanztypen zur Auswahl. Eine vollständige Liste finden Sie unter Amazon EC2 Instanztypen. Für diese Anleitung verwenden wir t1.micro, den günstigsten verfügbaren Instance-Typ. Als Nächstes bestimmen wir, welches AMI wir nutzen möchten. Wir werden ami-a9d09ed1 verwenden, das up-to-date Amazon Linux-AMI, das am meisten verfügbar war, als wir dieses Tutorial geschrieben haben. Von Zeit zu Zeit kann es neuere AMIs geben. Die jeweils neueste AMI-Version lässt sich mit folgenden Schritten ermitteln:

1. Öffnen Sie die [Amazon EC2 -Konsole](#).
2. Wählen Sie die Schaltfläche Launch Instance (Instance starten).
3. Im ersten Fenster werden die verfügbaren angezeigt. AMIs Die AMI-ID ist neben dem jeweiligen AMI-Titel aufgelistet. Alternativ können Sie die DescribeImages-API nutzen. Die Nutzung dieses Befehls wird in dieser Anleitung allerdings nicht behandelt.

Es gibt viele Wege zur Gebotsgestaltung für Spot-Instances. Eine gute Übersicht der unterschiedlichen Ansätze finden Sie im Video [Bieten für Spot-Instances](#). Allerdings beschreiben wir drei allgemeine Strategien für den Einstieg: Gebote, um sicherzustellen, dass die Kosten geringer sind als bei On-Demand-Preisen; Gebote basierend auf dem Wert der resultierenden Berechnung; Gebote, um Rechenkapazität so schnell wie möglich zu erwerben.

- Senkung der Kosten unter On-Demand Sie haben eine Stapelverarbeitungsaufgabe, die einige Stunden oder Tage laufen wird. Allerdings sind Sie flexibel, was den Start und Abschluss angeht. Sie möchten die Aufgabe nach Möglichkeit günstiger als mit On-Demand-Instances abschließen. Sie untersuchen den Spot-Price-Verlauf für Instance-Typen, indem Sie entweder die AWS-Managementkonsole oder die Amazon EC2 API verwenden. Weitere Informationen finden Sie unter [Anzeigen des Spot-Preisverlaufs](#). Nachdem Sie den Preisverlauf für Ihren gewünschten

Instance-Typ in einer bestimmten Availability Zone analysiert haben, gibt es zwei alternative Ansätze für Ihr Gebot:

- Sie könnten ein Gebot am oberen Ende der Skala für Spot-Preise abgeben (aber immer noch unter dem On-Demand-Preis) und voraussehen, dass Ihre einmalige Spot-Anforderung sehr wahrscheinlich erfüllt wird und die Instance lange genug aktiv bleibt, um die Aufgabe abzuschließen.
- Oder Sie können den Betrag, den Sie für Spot-Instances zu zahlen bereit sind, als % des On-Demand-Instance-Preises angeben und planen, viele Instances zu kombinieren, die im Laufe der Zeit über eine persistente Anforderungen gestartet wurden. Wenn der angegebene Preis überschritten wird, wird die Spot-Instance beendet. (Später in dieser Anleitung zeigen wir Ihnen, wie Sie diese Aufgabe automatisieren können.)
- Nicht mehr zahlen, als das Ergebnis einbringt Sie haben eine Aufgabe zur Datenverarbeitung, die ausgeführt werden soll. Sie kennen den Wert der Ergebnisse des Auftrags gut genug, um zu wissen, wie sich der Wert als Rechenkosten darstellen lässt. Sie analysieren den Spot-Preisverlauf Ihres Instance-Typs und entscheiden sich dann für einen Gebotspreis, der sicherstellt, dass die Kosten der Rechenzeit geringer sind als der Wert der Auftragsergebnisse. Sie erstellen ein persistentes Gebot und lassen es zwischenzeitlich laufen, sobald der Spot-Preis Ihr Gebot erreicht oder darunter sinkt.
- Schneller Erwerb von Rechenkapazität Sie haben einen plötzlichen, kurzfristigen Bedarf an zusätzlicher Kapazität, die von On-Demand-Instances nicht bereitgestellt werden kann. Sie analysieren den Spot-Preisverlauf Ihres Instance-Typs und bieten dann über dem höchsten bisherigen Preis. So sorgen Sie dafür, dass Ihre Anforderung mit hoher Wahrscheinlichkeit schnell erfüllt wird und bis zum Abschluss der Aufgabe läuft.

Nachdem Sie den Gebotspreis ausgewählt haben, können Sie eine Spot-Instance anfordern. Für diese Anleitung bieten wir den On-Demand-Preis (0,03 USD) und maximieren so die Chancen, dass das Gebot erfüllt wird. Sie können die Typen der verfügbaren Instances und die On-Demand-Preise für Instances auf der Seite mit den Amazon EC2 Preisen ermitteln. Für die Dauer der Ausführung der Instances zahlen Sie bei Spot-Instances den bei der Anforderung angegebenen Stundensatz. Die Spot-Instance-Preise werden festgelegt Amazon EC2 und auf der Grundlage langfristiger Trends bei Angebot und Nachfrage nach Spot-Instance-Kapazität schrittweise angepasst. Sie können auch den Betrag, den Sie für eine Spot-Instance zu zahlen bereit sind, als % des On-Demand-Instance-Preises angeben. Um eine Spot-Instance anzufordern, müssen Sie Ihre Anfrage einfach mit den zuvor ausgewählten Parametern erstellen. Als Erstes erstellen wir ein `RequestSpotInstanceRequest`-Objekt. Für das Anforderungsobjekt sind die Anzahl der zu startenden Instances sowie der

Gebotspreis nötig. Außerdem müssen Sie die `LaunchSpecification` für die Anforderung festlegen. Sie umfasst den Instance-Typ, die AMI-ID sowie die Sicherheitsgruppe, die Sie verwenden möchten. Sobald die Anforderung vorbereitet ist, rufen Sie die Methode `requestSpotInstances` des Objekts `AmazonEC2Client` auf. Das folgende Beispiel zeigt, wie Sie eine Spot-Instance anfordern.

```
// Create the AmazonEC2 client so we can call various APIs.
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

// Initializes a Spot Instance Request
RequestSpotInstancesRequest requestRequest = new RequestSpotInstancesRequest();

// Request 1 x t1.micro instance with a bid price of $0.03.
requestRequest.setSpotPrice("0.03");
requestRequest.setInstanceCount(Integer.valueOf(1));

// Setup the specifications of the launch. This includes the
// instance type (e.g. t1.micro) and the latest Amazon Linux
// AMI id available. Note, you should always use the latest
// Amazon Linux AMI id or another of your choosing.
LaunchSpecification launchSpecification = new LaunchSpecification();
launchSpecification.setImageId("ami-a9d09ed1");
launchSpecification.setInstanceType(InstanceType.T1Micro);

// Add the security group to the request.
ArrayList<String> securityGroups = new ArrayList<String>();
securityGroups.add("GettingStartedGroup");
launchSpecification.setSecurityGroups(securityGroups);

// Add the launch specifications to the request.
requestRequest.setLaunchSpecification(launchSpecification);

// Call the RequestSpotInstance API.
RequestSpotInstancesResult requestResult = ec2.requestSpotInstances(requestRequest);
```

Beim Ausführen dieses Codes wird eine neue Spot-Instance-Anforderung ausgeführt. Mit weiteren Optionen können Sie Spot-Anforderungen konfigurieren. Weitere Informationen finden Sie unter [Tutorial: Advanced Amazon EC2 Spot Request Management](#) oder im [RequestSpotInstances](#) Kurs in der AWS SDK für Java API-Referenz.

**Note**

Sie zahlen für Spot-Instances, die tatsächlich gestartet werden. Achten Sie also darauf, getätigte Anforderungen zu stornieren und gestartete Instances zu beenden, damit Ihnen keine unnötigen Kosten entstehen.

## Schritt 4: Ermitteln des Status Ihrer Spot-Instance-Anforderung

Als Nächstes erstellen wir einen Code, der darauf wartet, dass die Spot-Anforderung den Status "active" erreicht, bevor wir zum letzten Schritt wechseln. Um den Status unserer Spot-Anfrage zu ermitteln, fragen wir die Methode [describeSpotInstanceRequests](#) nach dem Status der Spot-Request-ID ab, die wir überwachen möchten.

Die in Schritt 2 erstellte Anforderungs-ID ist in der Antwort auf unsere requestSpotInstances-Anforderung enthalten. Der folgende Beispielcode zeigt, wie Sie Anfragen IDs aus der requestSpotInstances Antwort sammeln und sie verwenden, um eine ArrayList zu füllen.

```
// Call the RequestSpotInstance API.
RequestSpotInstancesResult requestResult = ec2.requestSpotInstances(requestRequest);
List<SpotInstanceRequest> requestResponses = requestResult.getSpotInstanceRequests();

// Setup an arraylist to collect all of the request ids we want to
// watch hit the running state.
ArrayList<String> spotInstanceRequestIds = new ArrayList<String>();

// Add all of the request ids to the hashset, so we can determine when they hit the
// active state.
for (SpotInstanceRequest requestResponse : requestResponses) {
    System.out.println("Created Spot Request:
"+requestResponse.getSpotInstanceRequestId());
    spotInstanceRequestIds.add(requestResponse.getSpotInstanceRequestId());
}
```

Rufen Sie zur Überwachung Ihrer Anforderungs-ID die Methode `describeSpotInstanceRequests` auf und ermitteln Sie so den Status der Anforderung. Warten Sie dann in einer Schleife, bis die Anforderung nicht mehr den Zustand "open" aufweist. Hinweis: Wir überprüfen hier, ob der Zustand ungleich "open" ist, anstatt etwa auf "active" zu überprüfen. Der Grund ist, dass die Anforderung direkt zu "closed" übergehen kann, wenn ein Problem mit den

Argumenten der Anforderung vorliegt. Im folgenden Codebeispiel sehen Sie im Detail, wie diese Aufgabe umgesetzt wird.

```
// Create a variable that will track whether there are any
// requests still in the open state.
boolean anyOpen;

do {
    // Create the describeRequest object with all of the request ids
    // to monitor (e.g. that we started).
    DescribeSpotInstanceRequestsRequest describeRequest = new
DescribeSpotInstanceRequestsRequest();
    describeRequest.setSpotInstanceRequestIds(spotInstanceRequestIds);

    // Initialize the anyOpen variable to false - which assumes there
    // are no requests open unless we find one that is still open.
    anyOpen=false;

    try {
        // Retrieve all of the requests we want to monitor.
        DescribeSpotInstanceRequestsResult describeResult =
ec2.describeSpotInstanceRequests(describeRequest);
        List<SpotInstanceRequest> describeResponses =
describeResult.getSpotInstanceRequests();

        // Look through each request and determine if they are all in
        // the active state.
        for (SpotInstanceRequest describeResponse : describeResponses) {
            // If the state is open, it hasn't changed since we attempted
            // to request it. There is the potential for it to transition
            // almost immediately to closed or cancelled so we compare
            // against open instead of active.
            if (describeResponse.getState().equals("open")) {
                anyOpen = true;
                break;
            }
        }
    } catch (AmazonServiceException e) {
        // If we have an exception, ensure we don't break out of
        // the loop. This prevents the scenario where there was
        // blip on the wire.
        anyOpen = true;
    }
}
```

```
try {
    // Sleep for 60 seconds.
    Thread.sleep(60*1000);
} catch (Exception e) {
    // Do nothing because it woke up early.
}
} while (anyOpen);
```

Nachdem dieser Code ausgeführt wird, ist Ihre Spot-Instance-Anforderung entweder abgeschlossen oder mit einem Fehler gescheitert, der auf dem Bildschirm angezeigt wird. In beiden Fällen können wir mit dem nächsten Schritt fortfahren, alle aktiven Anforderungen bereinigen und alle laufenden Instances beenden.

## Schritt 5: Bereinigen der Spot-Anforderungen und -Instances

Schließlich müssen wir unsere Anforderungen und Instances bereinigen. Es ist wichtig, sowohl ausstehende Anforderungen zu stornieren als auch etwaige Instances zu beenden. Wenn Sie nur die Anforderungen stornieren, werden Ihre Instances nicht beendet und Sie müssen weiter für sie zahlen. Wenn Sie die Instances beenden, können Ihre Spot-Anforderungen storniert werden. In einigen Fällen – etwa dann, wenn Sie persistente Gebote nutzen –, reicht das Beenden Ihrer Instances nicht aus, damit Ihre Anforderung nicht erneut erfüllt wird. Daher ist es eine bewährte Methode, sowohl die aktiven Gebote zu stornieren als auch alle laufenden Instances zu beenden.

Im folgenden Beispiel wird gezeigt, wie Sie Ihre Anforderungen stornieren.

```
try {
    // Cancel requests.
    CancelSpotInstanceRequestsRequest cancelRequest =
        new CancelSpotInstanceRequestsRequest(spotInstanceRequestIds);
    ec2.cancelSpotInstanceRequests(cancelRequest);
} catch (AmazonServiceException e) {
    // Write out any exceptions that may have occurred.
    System.out.println("Error cancelling instances");
    System.out.println("Caught Exception: " + e.getMessage());
    System.out.println("Reponse Status Code: " + e.getStatusCode());
    System.out.println("Error Code: " + e.getErrorCode());
    System.out.println("Request ID: " + e.getRequestId());
}
```

Zum Stornieren ausstehender Anforderungen brauchen Sie die jeweilige Instance-ID. Sie ist mit der Anforderung verknüpft, die sie gestartet hat. Im folgenden Codebeispiel ergänzen wir unseren Originalcode zur Überwachung der Instances um eine `ArrayList`. Darin speichern wir die Instance-ID aus der `describeInstance`-Antwort.

```
// Create a variable that will track whether there are any requests
// still in the open state.
boolean anyOpen;
// Initialize variables.
ArrayList<String> instanceIds = new ArrayList<String>();

do {
    // Create the describeRequest with all of the request ids to
    // monitor (e.g. that we started).
    DescribeSpotInstanceRequestsRequest describeRequest = new
DescribeSpotInstanceRequestsRequest();
    describeRequest.setSpotInstanceRequestIds(spotInstanceRequestIds);

    // Initialize the anyOpen variable to false, which assumes there
    // are no requests open unless we find one that is still open.
    anyOpen = false;

    try {
        // Retrieve all of the requests we want to monitor.
        DescribeSpotInstanceRequestsResult describeResult =
            ec2.describeSpotInstanceRequests(describeRequest);

        List<SpotInstanceRequest> describeResponses =
            describeResult.getSpotInstanceRequests();

        // Look through each request and determine if they are all
        // in the active state.
        for (SpotInstanceRequest describeResponse : describeResponses) {
            // If the state is open, it hasn't changed since we
            // attempted to request it. There is the potential for
            // it to transition almost immediately to closed or
            // cancelled so we compare against open instead of active.
            if (describeResponse.getState().equals("open")) {
                anyOpen = true; break;
            }
            // Add the instance id to the list we will
            // eventually terminate.
            instanceIds.add(describeResponse.getInstanceId());
        }
    }
}
```

```
    }
} catch (AmazonServiceException e) {
    // If we have an exception, ensure we don't break out
    // of the loop. This prevents the scenario where there
    // was blip on the wire.
    anyOpen = true;
}

try {
    // Sleep for 60 seconds.
    Thread.sleep(60*1000);
} catch (Exception e) {
    // Do nothing because it woke up early.
}
} while (anyOpen);
```

Beenden Sie mithilfe der Instance IDs, die in gespeichert ist `ArrayList`, alle laufenden Instances mithilfe des folgenden Codeausschnitts.

```
try {
    // Terminate instances.
    TerminateInstancesRequest terminateRequest = new
    TerminateInstancesRequest(instanceIds);
    ec2.terminateInstances(terminateRequest);
} catch (AmazonServiceException e) {
    // Write out any exceptions that may have occurred.
    System.out.println("Error terminating instances");
    System.out.println("Caught Exception: " + e.getMessage());
    System.out.println("Reponse Status Code: " + e.getStatusCode());
    System.out.println("Error Code: " + e.getErrorCode());
    System.out.println("Request ID: " + e.getRequestId());
}
```

## Zusammenfassung

Um all das zusammenzubringen, bieten wir einen objektorientierteren Ansatz, der die oben beschriebenen Schritte kombiniert: Initialisierung des EC2 Clients, Senden der Spot-Anfrage, Feststellung, wann sich die Spot-Anfragen nicht mehr im offenen Zustand befinden, und Bereinigen aller verbliebenen Spot-Anfragen und der zugehörigen Instances. Wir erstellen eine Klasse namens `Requests`, die diese Aktionen ausführt.

Zudem legen wir eine Klasse `GettingStartedApp` mit einer `main`-Methode an, in der wir die High-Level-Funktionsaufrufe durchführen. Insbesondere initialisieren wir das zuvor beschriebene `Requests`-Objekt. Wir senden die Spot-Instance-Anforderung. Anschließend warten wir, bis die Spot-Anforderung den Zustand "active" erreicht. Schließlich bereinigen wir die Anforderungen und Instances.

Der vollständige Quellcode für dieses Beispiel kann unter eingesehen oder heruntergeladen werden.

[GitHub](#)

Herzlichen Glückwunsch! Sie haben jetzt die Erste-Schritte-Anleitung zur Entwicklung von Spot-Instance-Software mit dem AWS SDK für Java abgeschlossen.

## Nächste Schritte

Fahren Sie mit [Tutorial: Erweitertes Amazon EC2 Spot-Anforderungsmanagement](#) fort.

## Tutorial: Erweitertes Amazon EC2 Spot-Anforderungsmanagement

Amazon EC2 Spot-Instances ermöglichen es Ihnen, Gebote für ungenutzte Amazon EC2 Kapazität abzugeben und diese Instances so lange laufen zu lassen, wie Ihr Gebot den aktuellen Spot-Preis übersteigt. Amazon EC2 ändert den Spot-Preis regelmäßig auf der Grundlage von Angebot und Nachfrage. Weitere Informationen zu Spot-Instances finden Sie unter [Spot-Instances](#) im Amazon EC2 Benutzerhandbuch für Linux-Instances.

## Voraussetzungen

Um dieses Tutorial verwenden zu können, müssen Sie das AWS SDK für Java installiert haben und die grundlegenden Installationsvoraussetzungen erfüllen. Weitere Informationen finden [Sie unter Einrichten](#) von AWS SDK für Java

## Einrichten Ihrer Anmeldeinformationen

Um mit der Verwendung dieses Codebeispiels zu beginnen, müssen Sie AWS Anmeldeinformationen einrichten. Anweisungen dazu finden Sie unter [AWS Zugangsdaten und Region für die Entwicklung einrichten](#).

**Note**

Wir empfehlen, dass Sie die Anmeldeinformationen eines IAM Benutzers verwenden, um diese Werte anzugeben. Weitere Informationen finden Sie unter [Registrieren für AWS und Erstellen eines IAM Benutzers](#).

Nachdem Sie Ihre Einstellungen eingerichtet haben, können Sie mit dem Beispiel-Code loslegen.

## Einrichten einer Sicherheitsgruppe

Eine Sicherheitsgruppe agiert als Firewall, die den zulässigen Verkehr zu und von einer Gruppe Instances steuert. Standardmäßig wird eine Instance ohne eine Sicherheitsgruppe gestartet. Sämtlicher eingehender IP-Datenverkehr auf allen TCP-Ports wird daher verweigert. Vor dem Absenden unserer Spot-Anforderung richten wir also eine Sicherheitsgruppe ein, die den nötigen Netzwerkverkehr zulässt. Für die Zwecke dieses Tutorials erstellen wir eine neue Sicherheitsgruppe namens "GettingStarted", die Secure Shell (SSH) -Verkehr von der IP-Adresse aus ermöglicht, von der aus Sie Ihre Anwendung ausführen. Zur Einrichtung einer neuen Sicherheitsgruppe sollten Sie das folgende Codebeispiel einschließen oder ausführen. Dadurch wird die Sicherheitsgruppe per Programm eingerichtet.

Nachdem wir ein AmazonEC2 Client-Objekt erstellt haben, erstellen wir ein `CreateSecurityGroupRequest` Objekt mit dem Namen "GettingStarted" und einer Beschreibung für die Sicherheitsgruppe. Anschließend wird die `ec2.createSecurityGroup`-API zum Erstellen der Gruppe aufgerufen.

Zum Aktivieren des Zugriffs auf die Gruppe erstellen wir ein `ipPermission`-Objekt, bei dem der IP-Adressbereich des Subnetzes auf die CIDR-Darstellung des Subnetzes des lokalen Computers festgelegt ist. Das Suffix `/10` bei der IP-Adresse zeigt das Subnetz für die angegebene IP-Adresse an. Anschließend konfigurieren wir auch das `ipPermission`-Objekt mit dem TCP-Protokoll und dem Port 22 (SSH). Im letzten Schritt wird `ec2.authorizeSecurityGroupIngress` mit dem Namen der Sicherheitsgruppe und dem `ipPermission`-Objekt aufgerufen.

(Der folgende Code entspricht unserem Code aus der ersten Anleitung.)

```
// Create the AmazonEC2Client object so we can call various APIs.
AmazonEC2 ec2 = AmazonEC2ClientBuilder.standard()
    .withCredentials(credentials)
    .build();
```

```
// Create a new security group.
try {
    CreateSecurityGroupRequest securityGroupRequest =
        new CreateSecurityGroupRequest("GettingStartedGroup",
            "Getting Started Security Group");
    ec2.createSecurityGroup(securityGroupRequest);
} catch (AmazonServiceException ase) {
    // Likely this means that the group is already created, so ignore.
    System.out.println(ase.getMessage());
}

String ipAddr = "0.0.0.0/0";

// Get the IP of the current host, so that we can limit the Security Group
// by default to the ip range associated with your subnet.
try {
    // Get IP Address
    InetAddress addr = InetAddress.getLocalHost();
    ipAddr = addr.getHostAddress()+"/10";
}
catch (UnknownHostException e) {
    // Fail here...
}

// Create a range that you would like to populate.
ArrayList<String> ipRanges = new ArrayList<String>();
ipRanges.add(ipAddr);

// Open up port 22 for TCP traffic to the associated IP from
// above (e.g. ssh traffic).
ArrayList<IpPermission> ipPermissions = new ArrayList<IpPermission> ();
IpPermission ipPermission = new IpPermission();
ipPermission.setIpProtocol("tcp");
ipPermission.setFromPort(new Integer(22));
ipPermission.setToPort(new Integer(22));
ipPermission.setIpRanges(ipRanges);
ipPermissions.add(ipPermission);

try {
    // Authorize the ports to the used.
    AuthorizeSecurityGroupIngressRequest ingressRequest =
        new AuthorizeSecurityGroupIngressRequest(
            "GettingStartedGroup", ipPermissions);
    ec2.authorizeSecurityGroupIngress(ingressRequest);
}
```

```
}  
catch (AmazonServiceException ase) {  
    // Ignore because this likely means the zone has already  
    // been authorized.  
    System.out.println(ase.getMessage());  
}
```

Sie können das gesamte Codebeispiel im `advanced.CreateSecurityGroupApp.java`-Codebeispiel einsehen. Hinweis: Sie müssen diese Anwendung nur einmal ausführen, um eine neue Sicherheitsgruppe zu erstellen.

### Note

Sie können die Sicherheitsgruppe auch mithilfe von AWS Toolkit for Eclipse erstellen. Weitere Informationen finden Sie AWS Cost Explorer im AWS Toolkit for Eclipse Benutzerhandbuch unter [Sicherheitsgruppen verwalten](#).

## Detaillierte Optionen für die Erstellung von Spot-Instance-Anforderungen

Wie wir im [Tutorial: Amazon EC2 Spot-Instances](#) erklärt haben, müssen Sie Ihre Anfrage mit einem Instance-Typ, einem Amazon Machine Image (AMI) und einem Höchstgebotspreis erstellen.

Als Erstes erstellen wir ein `RequestSpotInstanceRequest`-Objekt. Für das Anforderungsobjekt sind die Anzahl der gewünschten Instances sowie der Gebotspreis nötig. Außerdem müssen wir die `LaunchSpecification` für die Anforderung festlegen. Sie umfasst den Instance-Typ, die AMI-ID sowie die Sicherheitsgruppe, die Sie verwenden möchten. Nachdem die Anforderung vorbereitet ist, rufen wir die Methode `requestSpotInstances` des Objekts `AmazonEC2Client` auf. Es folgt ein Beispiel für die Anforderung einer Spot-Instance.

(Der folgende Code entspricht unserem Code aus der ersten Anleitung.)

```
// Create the AmazonEC2 client so we can call various APIs.  
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();  
  
// Initializes a Spot Instance Request  
RequestSpotInstancesRequest requestRequest = new RequestSpotInstancesRequest();  
  
// Request 1 x t1.micro instance with a bid price of $0.03.  
requestRequest.setSpotPrice("0.03");
```

```
requestRequest.setInstanceCount(Integer.valueOf(1));

// Set up the specifications of the launch. This includes the
// instance type (e.g. t1.micro) and the latest Amazon Linux
// AMI id available. Note, you should always use the latest
// Amazon Linux AMI id or another of your choosing.
LaunchSpecification launchSpecification = new LaunchSpecification();
launchSpecification.setImageId("ami-a9d09ed1");
launchSpecification.setInstanceType(InstanceType.T1Micro);

// Add the security group to the request.
ArrayList<String> securityGroups = new ArrayList<String>();
securityGroups.add("GettingStartedGroup");
launchSpecification.setSecurityGroups(securityGroups);

// Add the launch specification.
requestRequest.setLaunchSpecification(launchSpecification);

// Call the RequestSpotInstance API.
RequestSpotInstancesResult requestResult =
    ec2.requestSpotInstances(requestRequest);
```

## Persistente im Vergleich zu einmaligen Anforderungen

Beim Erstellen einer Spot-Instance-Anforderung können Sie mehrere optionale Parameter angeben. Die erste Option gibt an, ob Ihre Anforderung von einmaliger oder persistenter Natur sein soll. Standardmäßig handelt es sich um eine einmalige Anforderung. Eine einmalige Anforderung kann nur einmal erfüllt werden. Sind die angeforderten Instances beendet, wird die Anforderung geschlossen. Eine persistente Anforderung wird zur Erfüllung immer dann herangezogen, wenn für die gleiche Anforderung keine Spot-Instance ausgeführt wird. Geben Sie den Typ der Anforderung an, indem Sie einfach den "Type" auf der Spot-Anforderung festlegen. Dies lässt sich mit folgendem Code erreichen:

```
// Retrieves the credentials from an AWSCredentials.properties file.
AWSCredentials credentials = null;
try {
    credentials = new PropertiesCredentials(
        GettingStartedApp.class.getResourceAsStream("AwsCredentials.properties"));
}
catch (IOException e1) {
    System.out.println(
        "Credentials were not properly entered into AwsCredentials.properties.");
}
```

```
        System.out.println(e1.getMessage());
        System.exit(-1);
    }

    // Create the AmazonEC2 client so we can call various APIs.
    AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

    // Initializes a Spot Instance Request
    RequestSpotInstancesRequest requestRequest =
        new RequestSpotInstancesRequest();

    // Request 1 x t1.micro instance with a bid price of $0.03.
    requestRequest.setSpotPrice("0.03");
    requestRequest.setInstanceCount(Integer.valueOf(1));

    // Set the type of the bid to persistent.
    requestRequest.setType("persistent");

    // Set up the specifications of the launch. This includes the
    // instance type (e.g. t1.micro) and the latest Amazon Linux
    // AMI id available. Note, you should always use the latest
    // Amazon Linux AMI id or another of your choosing.
    LaunchSpecification launchSpecification = new LaunchSpecification();
    launchSpecification.setImageId("ami-a9d09ed1");
    launchSpecification.setInstanceType(InstanceType.T1Micro);

    // Add the security group to the request.
    ArrayList<String> securityGroups = new ArrayList<String>();
    securityGroups.add("GettingStartedGroup");
    launchSpecification.setSecurityGroups(securityGroups);

    // Add the launch specification.
    requestRequest.setLaunchSpecification(launchSpecification);

    // Call the RequestSpotInstance API.
    RequestSpotInstancesResult requestResult =
        ec2.requestSpotInstances(requestRequest);
```

## Einschränken der Dauer einer Anforderung

Außerdem können Sie optional festlegen, wie lange Ihre Anforderung gültig bleibt. Sie können für diesen Zeitraum eine Start- und Endzeit festlegen. Standardmäßig wird eine Spot-Anforderung zur Erfüllung von dem Augenblick ihrer Erstellung bis zu dem Zeitpunkt herangezogen, an dem sie

entweder erfüllt oder von Ihnen storniert wird. Allerdings können Sie die Gültigkeitsdauer bei Bedarf einschränken. Ein Beispiel dafür, wie Sie diesen Zeitraum angeben, wird im folgenden Code gezeigt:

```
// Create the AmazonEC2 client so we can call various APIs.
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

// Initializes a Spot Instance Request
RequestSpotInstancesRequest requestRequest = new RequestSpotInstancesRequest();

// Request 1 x t1.micro instance with a bid price of $0.03.
requestRequest.setSpotPrice("0.03");
requestRequest.setInstanceCount(Integer.valueOf(1));

// Set the valid start time to be two minutes from now.
Calendar cal = Calendar.getInstance();
cal.add(Calendar.MINUTE, 2);
requestRequest.setValidFrom(cal.getTime());

// Set the valid end time to be two minutes and two hours from now.
cal.add(Calendar.HOUR, 2);
requestRequest.setValidUntil(cal.getTime());

// Set up the specifications of the launch. This includes
// the instance type (e.g. t1.micro)

// and the latest Amazon Linux AMI id available.
// Note, you should always use the latest Amazon
// Linux AMI id or another of your choosing.
LaunchSpecification launchSpecification = new LaunchSpecification();
launchSpecification.setImageId("ami-a9d09ed1");
launchSpecification.setInstanceType("t1.micro");

// Add the security group to the request.
ArrayList<String> securityGroups = new ArrayList<String>();
securityGroups.add("GettingStartedGroup");
launchSpecification.setSecurityGroups(securityGroups);

// Add the launch specification.
requestRequest.setLaunchSpecification(launchSpecification);

// Call the RequestSpotInstance API.
RequestSpotInstancesResult requestResult = ec2.requestSpotInstances(requestRequest);
```

## Gruppieren Sie Ihre Amazon EC2 Spot-Instance-Anfragen

Auf Wunsch können Sie Spot-Instance-Anforderungen auf verschiedene Weise gruppieren. Beachten Sie die Vorteile der Nutzung von Start-, Availability Zone- und Platzierungsgruppen.

Wenn Sie sichergehen möchten, dass Ihre Spot-Instances gleichzeitig gestartet und beendet werden, können Sie eine Startgruppe nutzen. Eine Startgruppe ist eine Bezeichnung, die einige Gebote gruppiert. Alle Instances in einer Startgruppe werden zusammen gestartet und beendet. Hinweis: Wurden Instances in einer Startgruppe bereits erfüllt, gibt es keine Garantie dafür, dass neu in der gleichen Startgruppe gestarteten Instances ebenfalls erfüllt werden. Ein Beispiel dafür, wie Sie eine Startgruppe festlegen können, wird im folgenden Code gezeigt:

```
// Create the AmazonEC2 client so we can call various APIs.
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

// Initializes a Spot Instance Request
RequestSpotInstancesRequest requestRequest = new RequestSpotInstancesRequest();

// Request 5 x t1.micro instance with a bid price of $0.03.
requestRequest.setSpotPrice("0.03");
requestRequest.setInstanceCount(Integer.valueOf(5));

// Set the launch group.
requestRequest.setLaunchGroup("ADVANCED-DEMO-LAUNCH-GROUP");

// Set up the specifications of the launch. This includes
// the instance type (e.g. t1.micro) and the latest Amazon Linux
// AMI id available. Note, you should always use the latest
// Amazon Linux AMI id or another of your choosing.
LaunchSpecification launchSpecification = new LaunchSpecification();
launchSpecification.setImageId("ami-a9d09ed1");
launchSpecification.setInstanceType(InstanceType.T1Micro);

// Add the security group to the request.
ArrayList<String> securityGroups = new ArrayList<String>();
securityGroups.add("GettingStartedGroup");
launchSpecification.setSecurityGroups(securityGroups);

// Add the launch specification.
requestRequest.setLaunchSpecification(launchSpecification);

// Call the RequestSpotInstance API.
```

```
RequestSpotInstancesResult requestResult =  
    ec2.requestSpotInstances(requestRequest);
```

Wenn Sie sicherstellen möchten, dass alle Instances in einer Anforderung in derselben Availability Zone gestartet werden, und es nicht relevant ist, in welcher Zone, können Sie Availability Zone-Gruppen nutzen. Eine Availability Zone-Gruppe ist eine Bezeichnung, die eine Gruppe von Instances in derselben Availability Zone gruppiert. Alle Instances mit der gleichen Availability Zone-Gruppe, die gleichzeitig erfüllt werden, starten in derselben Availability Zone. Ein Beispiel für die Einrichtung einer Availability Zone-Gruppe finden Sie hier:

```
// Create the AmazonEC2 client so we can call various APIs.  
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();  
  
// Initializes a Spot Instance Request  
RequestSpotInstancesRequest requestRequest = new RequestSpotInstancesRequest();  
  
// Request 5 x t1.micro instance with a bid price of $0.03.  
requestRequest.setSpotPrice("0.03");  
requestRequest.setInstanceCount(Integer.valueOf(5));  
  
// Set the availability zone group.  
requestRequest.setAvailabilityZoneGroup("ADVANCED-DEMO-AZ-GROUP");  
  
// Set up the specifications of the launch. This includes the instance  
// type (e.g. t1.micro) and the latest Amazon Linux AMI id available.  
// Note, you should always use the latest Amazon Linux AMI id or another  
// of your choosing.  
LaunchSpecification launchSpecification = new LaunchSpecification();  
launchSpecification.setImageId("ami-a9d09ed1");  
launchSpecification.setInstanceType(InstanceType.T1Micro);  
  
// Add the security group to the request.  
ArrayList<String> securityGroups = new ArrayList<String>();  
securityGroups.add("GettingStartedGroup");  
launchSpecification.setSecurityGroups(securityGroups);  
  
// Add the launch specification.  
requestRequest.setLaunchSpecification(launchSpecification);  
  
// Call the RequestSpotInstance API.  
RequestSpotInstancesResult requestResult =  
    ec2.requestSpotInstances(requestRequest);
```

Sie können eine Availability Zone angeben, die Sie für Ihre Spot-Instances nutzen möchten. Das folgende Codebeispiel zeigt, wie Sie eine Availability Zone festlegen.

```
// Create the AmazonEC2 client so we can call various APIs.
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

// Initializes a Spot Instance Request
RequestSpotInstances requestRequest = new RequestSpotInstancesRequest();

// Request 1 x t1.micro instance with a bid price of $0.03.
requestRequest.setSpotPrice("0.03");
requestRequest.setInstanceCount(Integer.valueOf(1));

// Set up the specifications of the launch. This includes the instance
// type (e.g. t1.micro) and the latest Amazon Linux AMI id available.
// Note, you should always use the latest Amazon Linux AMI id or another
// of your choosing.
LaunchSpecification launchSpecification = new LaunchSpecification();
launchSpecification.setImageId("ami-a9d09ed1");
launchSpecification.setInstanceType(InstanceType.T1Micro);

// Add the security group to the request.
ArrayList<String> securityGroups = new ArrayList<String>();
securityGroups.add("GettingStartedGroup");
launchSpecification.setSecurityGroups(securityGroups);

// Set up the availability zone to use. Note we could retrieve the
// availability zones using the ec2.describeAvailabilityZones() API. For
// this demo we will just use us-east-1a.
SpotPlacement placement = new SpotPlacement("us-east-1b");
launchSpecification.setPlacement(placement);

// Add the launch specification.
requestRequest.setLaunchSpecification(launchSpecification);

// Call the RequestSpotInstance API.
RequestSpotInstancesResult requestResult =
    ec2.requestSpotInstances(requestRequest);
```

Schließlich können Sie eine Platzierungsgruppe angeben, wenn Sie High Performance Computing (HPC)-Spot-Instances nutzen, etwa Cluster Compute-Instances oder Cluster-GPU-Instances.

Platzierungsgruppen sorgen für eine niedrigere Latenz und eine hohe Bandbreitenkonnectivität zwischen den Instances. Ein Beispiel für die Einrichtung einer Platzierungsgruppe finden Sie hier:

```
// Create the AmazonEC2 client so we can call various APIs.
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

// Initializes a Spot Instance Request
RequestSpotInstancesRequest requestRequest = new RequestSpotInstancesRequest();

// Request 1 x t1.micro instance with a bid price of $0.03.
requestRequest.setSpotPrice("0.03");
requestRequest.setInstanceCount(Integer.valueOf(1));

// Set up the specifications of the launch. This includes the instance
// type (e.g. t1.micro) and the latest Amazon Linux AMI id available.
// Note, you should always use the latest Amazon Linux AMI id or another
// of your choosing.

LaunchSpecification launchSpecification = new LaunchSpecification();
launchSpecification.setImageId("ami-a9d09ed1");
launchSpecification.setInstanceType(InstanceType.T1Micro);

// Add the security group to the request.
ArrayList<String> securityGroups = new ArrayList<String>();
securityGroups.add("GettingStartedGroup");
launchSpecification.setSecurityGroups(securityGroups);

// Set up the placement group to use with whatever name you desire.
// For this demo we will just use "ADVANCED-DEMO-PLACEMENT-GROUP".
SpotPlacement placement = new SpotPlacement();
placement.setGroupName("ADVANCED-DEMO-PLACEMENT-GROUP");
launchSpecification.setPlacement(placement);

// Add the launch specification.
requestRequest.setLaunchSpecification(launchSpecification);

// Call the RequestSpotInstance API.
RequestSpotInstancesResult requestResult =
    ec2.requestSpotInstances(requestRequest);
```

Alle Parameter in diesem Abschnitt sind optional. Es ist auch wichtig zu wissen, dass die meisten dieser Parameter — mit Ausnahme der Tatsache, ob es sich bei Ihrem Gebot um ein einmaliges oder ein dauerhaftes Gebot handelt — die Wahrscheinlichkeit verringern

können, dass das Gebot erfüllt wird. Daher ist es wichtig, diese Optionen nur bei Bedarf zu nutzen. Alle obigen Codebeispiele sind in einem langen Codebeispiel kombiniert, das in der `com.amazonaws.codesamples.advanced.InlineGettingStartedCodeSampleApp.java`-Klasse zu finden ist.

## So bleibt eine Stammpartition nach einer Unterbrechung oder Beendigung erhalten

Eine der einfachsten Möglichkeiten, mit Unterbrechungen Ihrer Spot-Instances umzugehen, besteht darin, sicherzustellen, dass Ihre Daten regelmäßig auf ein Amazon Elastic Block Store (Amazon EBS) -Volume (Amazon) überprüft werden. Durch das Setzen von Prüfpunkten in regelmäßigen Abständen verlieren Sie im Falle einer Unterbrechung nur die seit dem letzten Prüfpunkt erstellten Daten (angenommen, zwischenzeitlich wurden keine anderen idempotenten Aktionen ausgeführt). Um diesen Prozess zu vereinfachen, können Sie Ihre Spot-Anforderung so konfigurieren, dass Ihre Stammpartition bei Unterbrechungen oder Beendigungen nicht gelöscht wird. Im folgenden Beispiel haben wir neuen Code eingefügt, der zeigt, wie sich dieses Szenario umsetzen lässt.

Im hinzugefügten Code erstellen wir ein `BlockDeviceMapping` Objekt und setzen es mit einem Amazon EBS Objekt verknüpft Amazon Elastic Block Store (Amazon EBS), das wir so konfiguriert haben, dass es gelöscht wird, wenn die Spot-Instance beendet wird. Wir fügen dies dann `BlockDeviceMapping` zu den `ArrayList` Mappings hinzu, die wir in die Startspezifikation aufnehmen.

```
// Retrieves the credentials from an AWSCredentials.properties file.
AWSCredentials credentials = null;
try {
    credentials = new PropertiesCredentials(
        GettingStartedApp.class.getResourceAsStream("AwsCredentials.properties"));
}
catch (IOException e1) {
    System.out.println(
        "Credentials were not properly entered into AwsCredentials.properties.");
    System.out.println(e1.getMessage());
    System.exit(-1);
}

// Create the AmazonEC2 client so we can call various APIs.
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

// Initializes a Spot Instance Request
RequestSpotInstancesRequest requestRequest = new RequestSpotInstancesRequest();
```

```
// Request 1 x t1.micro instance with a bid price of $0.03.
requestRequest.setSpotPrice("0.03");
requestRequest.setInstanceCount(Integer.valueOf(1));

// Set up the specifications of the launch. This includes the instance
// type (e.g. t1.micro) and the latest Amazon Linux AMI id available.
// Note, you should always use the latest Amazon Linux AMI id or another
// of your choosing.
LaunchSpecification launchSpecification = new LaunchSpecification();
launchSpecification.setImageId("ami-a9d09ed1");
launchSpecification.setInstanceType(InstanceType.T1Micro);

// Add the security group to the request.
ArrayList<String> securityGroups = new ArrayList<String>();
securityGroups.add("GettingStartedGroup");
launchSpecification.setSecurityGroups(securityGroups);

// Create the block device mapping to describe the root partition.
BlockDeviceMapping blockDeviceMapping = new BlockDeviceMapping();
blockDeviceMapping.setDeviceName("/dev/sda1");

// Set the delete on termination flag to false.
EbsBlockDevice ebs = new EbsBlockDevice();
ebs.setDeleteOnTermination(Boolean.FALSE);
blockDeviceMapping.setEbs(ebs);

// Add the block device mapping to the block list.
ArrayList<BlockDeviceMapping> blockList = new ArrayList<BlockDeviceMapping>();
blockList.add(blockDeviceMapping);

// Set the block device mapping configuration in the launch specifications.
launchSpecification.setBlockDeviceMappings(blockList);

// Add the launch specification.
requestRequest.setLaunchSpecification(launchSpecification);

// Call the RequestSpotInstance API.
RequestSpotInstancesResult requestResult =
    ec2.requestSpotInstances(requestRequest);
```

Wenn Sie dieses Volume beim Startup erneut an Ihre Instance anfügen möchten, können Sie auch die Einstellungen für Blockgerät-Zuweisung verwenden. Wenn Sie eine Nicht-Root-Partition

angehängt haben, können Sie alternativ die Amazon EBS Amazon-Volumes angeben, die Sie an Ihre Spot-Instance anhängen möchten, nachdem sie wieder aufgenommen wurde. Geben Sie dazu einfach eine Snapshot-ID in Ihrem `EbsBlockDevice` und einen alternativen Gerätenamen in Ihren `BlockDeviceMapping`-Objekten an. Durch die Nutzung von Blockgerät-Zuweisungen lässt sich die Instance einfacher starten.

Wenn Sie die Stammpartition verwenden, um Prüfpunkte für Ihre wichtigen Daten anzulegen, können Sie auf diese Weise die Wahrscheinlichkeit der Unterbrechung Ihrer Instances im Griff behalten. Weitere Methoden zum Umgang mit der Wahrscheinlichkeit von Unterbrechungen finden Sie im Video [Managing Interruption](#).

## So markieren Sie Spot-Anforderungen und -Instances

Das Hinzufügen von Tags zu Amazon EC2 Ressourcen kann die Verwaltung Ihrer Cloud-Infrastruktur vereinfachen. Tags sind ein Typ von Metadaten, der verwendet werden kann, um benutzerfreundliche Namen zu erstellen, die Durchsuchbarkeit zu optimieren und die Koordination zwischen mehreren Benutzern zu verbessern. Sie können Tags auch zur Automatisierung von Skripts und Teilen Ihrer Prozesse nutzen. Weitere Informationen zum Taggen von Amazon EC2 Ressourcen finden Sie [unter Verwenden von Tags](#) im Amazon EC2 Benutzerhandbuch für Linux-Instances.

### Markieren von -Anforderungen

Zum Hinzufügen von Tags zu Ihren Spot-Anforderungen müssen Sie sie markieren, nachdem sie angefordert wurden. Der Rückgabewert von `requestSpotInstances()` stellt Ihnen ein [RequestSpotInstancesResult](#)-Objekt zur Verfügung, mit dem Sie die Spot-Anfrage IDs für das Tagging abrufen können:

```
// Call the RequestSpotInstance API.
RequestSpotInstancesResult requestResult = ec2.requestSpotInstances(requestRequest);
List<SpotInstanceRequest> requestResponses = requestResult.getSpotInstanceRequests();

// A list of request IDs to tag
ArrayList<String> spotInstanceRequestIds = new ArrayList<String>();

// Add the request ids to the hashset, so we can determine when they hit the
// active state.
for (SpotInstanceRequest requestResponse : requestResponses) {
    System.out.println("Created Spot Request:
"+requestResponse.getSpotInstanceRequestId());
    spotInstanceRequestIds.add(requestResponse.getSpotInstanceRequestId());
}
```

Sobald Sie die haben IDs, können Sie die Anfragen taggen, indem Sie sie IDs zu einer hinzufügen [CreateTagsRequest](#) und die `createTags()` Methode des Amazon EC2 Clients aufrufen:

```
// The list of tags to create
ArrayList<Tag> requestTags = new ArrayList<Tag>();
requestTags.add(new Tag("keyname1","value1"));

// Create the tag request
CreateTagsRequest createTagsRequest_requests = new CreateTagsRequest();
createTagsRequest_requests.setResources(spotInstanceRequestIds);
createTagsRequest_requests.setTags(requestTags);

// Tag the spot request
try {
    ec2.createTags(createTagsRequest_requests);
}
catch (AmazonServiceException e) {
    System.out.println("Error terminating instances");
    System.out.println("Caught Exception: " + e.getMessage());
    System.out.println("Reponse Status Code: " + e.getStatusCode());
    System.out.println("Error Code: " + e.getErrorCode());
    System.out.println("Request ID: " + e.getRequestId());
}
```

## Markieren von Instances

Ähnlich wie bei Spot-Anforderungen selbst können Sie eine Instance erst nach ihrer Erstellung markieren. Instances werden erstellt, sobald die Spot-Anforderung erfüllt wurde (d. h., wenn sie nicht mehr den Status offen hat).

Sie können den Status Ihrer Anfragen überprüfen, indem Sie die `describeSpotInstanceRequests()` Methode des Amazon EC2 Clients mit einem [DescribeSpotInstanceRequestsRequest](#) Objekt aufrufen. Das zurückgegebene [DescribeSpotInstanceRequestsResult](#) Objekt enthält eine Liste von [SpotInstanceRequest](#) Objekten, mit denen Sie den Status Ihrer Spot-Anfragen abfragen und deren Instanz abrufen können, IDs sobald sie sich nicht mehr im geöffneten Zustand befinden.

Sobald die Spot-Anforderung nicht mehr offen ist, können Sie ihre Instance-ID vom `SpotInstanceRequest`-Objekt erhalten, indem Sie dessen `getInstanceId()`-Methode aufrufen.

```
boolean anyOpen; // tracks whether any requests are still open
```

```
// a list of instances to tag.
ArrayList<String> instanceIds = new ArrayList<String>();

do {
    DescribeSpotInstanceRequestsRequest describeRequest =
        new DescribeSpotInstanceRequestsRequest();
    describeRequest.setSpotInstanceRequestIds(spotInstanceRequestIds);

    anyOpen=false; // assume no requests are still open

    try {
        // Get the requests to monitor
        DescribeSpotInstanceRequestsResult describeResult =
            ec2.describeSpotInstanceRequests(describeRequest);

        List<SpotInstanceRequest> describeResponses =
            describeResult.getSpotInstanceRequests();

        // are any requests open?
        for (SpotInstanceRequest describeResponse : describeResponses) {
            if (describeResponse.getState().equals("open")) {
                anyOpen = true;
                break;
            }
            // get the corresponding instance ID of the spot request
            instanceIds.add(describeResponse.getInstanceId());
        }
    }
    catch (AmazonServiceException e) {
        // Don't break the loop due to an exception (it may be a temporary issue)
        anyOpen = true;
    }

    try {
        Thread.sleep(60*1000); // sleep 60s.
    }
    catch (Exception e) {
        // Do nothing if the thread woke up early.
    }
} while (anyOpen);
```

Jetzt können Sie die zurückgegebenen Instances markieren:

```
// Create a list of tags to create
ArrayList<Tag> instanceTags = new ArrayList<Tag>();
instanceTags.add(new Tag("keyname1","value1"));

// Create the tag request
CreateTagsRequest createTagsRequest_instances = new CreateTagsRequest();
createTagsRequest_instances.setResources(instanceIds);
createTagsRequest_instances.setTags(instanceTags);

// Tag the instance
try {
    ec2.createTags(createTagsRequest_instances);
}
catch (AmazonServiceException e) {
    // Write out any exceptions that may have occurred.
    System.out.println("Error terminating instances");
    System.out.println("Caught Exception: " + e.getMessage());
    System.out.println("Reponse Status Code: " + e.getStatusCode());
    System.out.println("Error Code: " + e.getErrorCode());
    System.out.println("Request ID: " + e.getRequestId());
}
```

## Stornieren von Spot-Anforderungen und Beenden von Instances

### Stornieren einer Spot-Anforderung

Um eine Spot-Instance-Anfrage zu stornieren, rufen Sie den Amazon EC2 Client mit einem [CancelSpotInstanceRequestsRequest](#) Objekt `cancelSpotInstanceRequests` auf.

```
try {
    CancelSpotInstanceRequestsRequest cancelRequest = new
    CancelSpotInstanceRequestsRequest(spotInstanceRequestIds);
    ec2.cancelSpotInstanceRequests(cancelRequest);
} catch (AmazonServiceException e) {
    System.out.println("Error cancelling instances");
    System.out.println("Caught Exception: " + e.getMessage());
    System.out.println("Reponse Status Code: " + e.getStatusCode());
    System.out.println("Error Code: " + e.getErrorCode());
    System.out.println("Request ID: " + e.getRequestId());
}
```

## Beenden von Spot-Instances

Sie können alle laufenden Spot-Instances beenden, indem Sie sie IDs an die `terminateInstances()` Methode des Amazon EC2 Clients übergeben.

```
try {
    TerminateInstancesRequest terminateRequest = new
    TerminateInstancesRequest(instanceIds);
    ec2.terminateInstances(terminateRequest);
} catch (AmazonServiceException e) {
    System.out.println("Error terminating instances");
    System.out.println("Caught Exception: " + e.getMessage());
    System.out.println("Reponse Status Code: " + e.getStatusCode());
    System.out.println("Error Code: " + e.getErrorCode());
    System.out.println("Request ID: " + e.getRequestId());
}
```

## Zusammenfassung

Fassen wir zusammen: Wir bieten einen eher objektorientierten Ansatz, der die in dieser Anleitung gezeigten Schritte in einer einfach einsetzbaren Klasse kombiniert. Wir instanzieren eine Klasse namens `Requests`, die diese Aktionen ausführt. Zudem legen wir eine Klasse `GettingStartedApp` mit einer `main`-Methode an, in der wir die High-Level-Funktionsaufrufe durchführen.

Der vollständige Quellcode für dieses Beispiel kann unter eingesehen oder heruntergeladen werden [GitHub](#).

Herzlichen Glückwunsch! Sie haben jetzt die Anleitung "Erweiterte Anforderungsfunktionen" zur Entwicklung von Spot-Instance-Software mit dem AWS SDK für Java abgeschlossen.

## Amazon EC2 Instanzen verwalten

### Erstellen einer Instance

Erstellen Sie eine neue Amazon EC2 Instance, indem Sie die `runInstances` Methode des EC2 Amazon-Clients aufrufen und ihr ein [RunInstancesRequest](#) zu verwendendes [Amazon Machine Image \(AMI\)](#) und einen [Instance-Typ](#) zur Verfügung stellen.

### Importe

```
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
```

```
import com.amazonaws.services.ec2.model.InstanceType;
import com.amazonaws.services.ec2.model.RunInstancesRequest;
import com.amazonaws.services.ec2.model.RunInstancesResult;
import com.amazonaws.services.ec2.model.Tag;
```

## Code

```
RunInstancesRequest run_request = new RunInstancesRequest()
    .withImageId(ami_id)
    .withInstanceType(InstanceType.T1Micro)
    .withMaxCount(1)
    .withMinCount(1);

RunInstancesResult run_response = ec2.runInstances(run_request);

String reservation_id =
    run_response.getReservation().getInstances().get(0).getInstanceId();
```

Siehe [vollständiges Beispiel](#).

## Starten einer Instance

Um eine Amazon EC2 Instance zu starten, rufen Sie die `startInstances` Methode des EC2 Amazon-Clients auf und geben ihr eine, die die ID der zu startenden Instance [StartInstancesRequest](#) enthält.

## Importe

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.StartInstancesRequest;
```

## Code

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

StartInstancesRequest request = new StartInstancesRequest()
    .withInstanceIds(instance_id);

ec2.startInstances(request);
```

Siehe [vollständiges Beispiel](#).

## Anhalten einer Instance

Um eine Amazon EC2 Instance zu stoppen, rufen Sie die `stopInstances` Methode des EC2 Amazon-Clients auf und geben ihr eine, die die ID der zu stoppenden Instance [StopInstancesRequest](#) enthält.

### Importe

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.StopInstancesRequest;
```

### Code

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

StopInstancesRequest request = new StopInstancesRequest()
    .withInstanceIds(instance_id);

ec2.stopInstances(request);
```

Siehe [vollständiges Beispiel](#).

## Neustarten einer Instance

Um eine Amazon EC2 Instance neu zu starten, rufen Sie die `rebootInstances` Methode des EC2 Amazon-Clients auf und geben ihr eine, die die ID der Instance [RebootInstancesRequest](#) enthält, die neu gestartet werden soll.

### Importe

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.RebootInstancesRequest;
import com.amazonaws.services.ec2.model.RebootInstancesResult;
```

### Code

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

RebootInstancesRequest request = new RebootInstancesRequest()
```

```
.withInstanceIds(instance_id);

RebootInstancesResult response = ec2.rebootInstances(request);
```

Siehe [vollständiges Beispiel](#).

## Beschreiben von Instances

Um Ihre Instances aufzulisten, erstellen Sie eine `describeInstances` Methode des EC2 Amazon-Clients [DescribeInstancesRequest](#) und rufen Sie sie auf. Es wird ein [DescribeInstancesResult](#) Objekt zurückgegeben, mit dem Sie die Amazon EC2 Instances 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, sollten Sie zuerst die `getReservations` method, and then call `getInstances`-Methode der `DescribeInstancesResult`-Klasse für jedes zurückgegebene [Reservation](#)-Objekt aufrufen.

### Importe

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
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.Reservation;
```

### Code

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();
boolean done = false;

DescribeInstancesRequest request = new DescribeInstancesRequest();
while(!done) {
    DescribeInstancesResult response = ec2.describeInstances(request);

    for(Reservation reservation : response.getReservations()) {
        for(Instance instance : reservation.getInstances()) {
            System.out.printf(
                "Found instance with id %s, " +
                "AMI %s, " +
                "type %s, " +
```

```
        "state %s " +
        "and monitoring state %s",
        instance.getInstanceId(),
        instance.getImageId(),
        instance.getInstanceType(),
        instance.getState().getName(),
        instance.getMonitoring().getState());
    }
}

request.setNextToken(response.getNextToken());

if(response.getNextToken() == null) {
    done = true;
}
}
```

Die Ergebnisse werden seitenweise zurückgegeben. Sie können die weiteren Ergebnisse abrufen, indem Sie den von der `getNextToken`-Methode des Rückgabeobjekts zurückgegebenen Wert an die `setNextToken`-Methode des Original-Anforderungsobjekts übergeben. Verwenden Sie dann das gleiche Anforderungsobjekt für den nächsten Aufruf von `describeInstances`.

Siehe [vollständiges Beispiel](#).

## Überwachung einer Instance

Sie können verschiedene Aspekte Ihrer Amazon EC2 Instances überwachen, z. B. die CPU- und Netzwerkauslastung, den verfügbaren Arbeitsspeicher und den verbleibenden Festplattenspeicher. Weitere Informationen zur Instanzüberwachung finden Sie unter [Überwachung Amazon EC2](#) im Amazon EC2 Benutzerhandbuch für Linux-Instances.

Um mit der Überwachung einer Instance zu beginnen, müssen Sie eine [MonitorInstancesRequest](#) mit der ID der zu überwachenden Instance erstellen und sie an die `monitorInstances` Methode des EC2 Amazon-Clients übergeben.

### Importe

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.MonitorInstancesRequest;
```

### Code

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

MonitorInstancesRequest request = new MonitorInstancesRequest()
    .withInstanceIds(instance_id);

ec2.monitorInstances(request);
```

Siehe [vollständiges Beispiel](#).

## Anhalten der Instance-Überwachung

Um die Überwachung einer Instance zu beenden, erstellen Sie eine [UnmonitorInstancesRequest](#) mit der ID der Instance, deren Überwachung beendet werden soll, und übergeben Sie sie an die `unmonitorInstances` Methode des EC2 Amazon-Clients.

### Importe

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.UnmonitorInstancesRequest;
```

### Code

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

UnmonitorInstancesRequest request = new UnmonitorInstancesRequest()
    .withInstanceIds(instance_id);

ec2.unmonitorInstances(request);
```

Siehe [vollständiges Beispiel](#).

## 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

## Verwendung von Elastic IP-Adressen in Amazon EC2

EC2-Classic geht in den Ruhestand

### Warning

Wir gehen in den Ruhestand EC2 -Classic am 15. August 2022. Wir empfehlen Ihnen, von EC2 -Classic zu einer VPC zu migrieren. Weitere Informationen finden Sie im Blogbeitrag [EC2-Classic-Classic-Classic Networking is Retiring](#) — So bereiten Sie sich vor.

## Zuweisen einer Elastic IP-Adresse

Um eine Elastic IP-Adresse zu verwenden, verknüpfen Sie sie zuerst mit Ihrem Konto und anschließend mit Ihrer Instance oder Netzwerkschnittstelle.

Um eine Elastic IP-Adresse zuzuweisen, rufen Sie die `allocateAddress` Methode des EC2 Amazon-Clients mit einem [AllocateAddressRequest](#) Objekt auf, das den Netzwerktyp (klassisch EC2 oder VPC) enthält.

Die zurückgegebene Datei [AllocateAddressResult](#) enthält eine Zuweisungs-ID, mit der Sie die Adresse einer Instance zuordnen können, indem Sie die Zuweisungs-ID und die Instance-ID in a [AssociateAddressRequest](#) an die `associateAddress` Methode des EC2 Amazon-Clients übergeben.

## Importe

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.AllocateAddressRequest;
import com.amazonaws.services.ec2.model.AllocateAddressResult;
import com.amazonaws.services.ec2.model.AssociateAddressRequest;
import com.amazonaws.services.ec2.model.AssociateAddressResult;
import com.amazonaws.services.ec2.model.DomainType;
```

## Code

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

AllocateAddressRequest allocate_request = new AllocateAddressRequest()
    .withDomain(DomainType.Vpc);

AllocateAddressResult allocate_response =
    ec2.allocateAddress(allocate_request);

String allocation_id = allocate_response.getAllocationId();

AssociateAddressRequest associate_request =
    new AssociateAddressRequest()
        .withInstanceId(instance_id)
        .withAllocationId(allocation_id);

AssociateAddressResult associate_response =
    ec2.associateAddress(associate_request);
```

Siehe [vollständiges Beispiel](#).

## Beschreiben von Elastic IP-Adressen

Um die Elastic IP-Adressen aufzulisten, die Ihrem Konto zugewiesen sind, rufen Sie die `describeAddresses` Methode des EC2 Amazon-Clients auf. Sie gibt eine zurück [DescribeAddressesResult](#), mit der Sie eine Liste von [Address-Objekten](#) abrufen können, die die Elastic IP-Adressen in Ihrem Konto repräsentieren.

### Importe

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.Address;
import com.amazonaws.services.ec2.model.DescribeAddressesResult;
```

### Code

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

DescribeAddressesResult response = ec2.describeAddresses();

for(Address address : response.getAddresses()) {
```

```
System.out.printf(
    "Found address with public IP %s, " +
    "domain %s, " +
    "allocation id %s " +
    "and NIC id %s",
    address.getPublicIp(),
    address.getDomain(),
    address.getAllocationId(),
    address.getNetworkInterfaceId());
}
```

Siehe [vollständiges Beispiel](#).

## Freigeben einer Elastic IP-Adresse

Um eine Elastic IP-Adresse freizugeben, rufen Sie die `releaseAddress` Methode des EC2 Amazon-Clients auf und übergeben Sie ihr eine, die die Zuweisungs-ID der Elastic IP-Adresse [ReleaseAddressRequest](#) enthält, die Sie freigeben möchten.

### Importe

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.ReleaseAddressRequest;
import com.amazonaws.services.ec2.model.ReleaseAddressResult;
```

### Code

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

ReleaseAddressRequest request = new ReleaseAddressRequest()
    .withAllocationId(alloc_id);

ReleaseAddressResult response = ec2.releaseAddress(request);
```

Nachdem Sie eine Elastic IP-Adresse freigegeben haben, wird sie für den AWS IP-Adresspool freigegeben und steht Ihnen danach möglicherweise nicht mehr zur Verfügung. Achten Sie darauf, die DNS-Datensätze sowie alle Server und Geräte zu aktualisieren, die mit der Adresse kommunizieren. Wenn Sie versuchen, eine Elastic IP-Adresse freizugeben, die Sie bereits veröffentlicht haben, erhalten Sie eine `AuthFailure` Fehlermeldung, wenn die Adresse bereits einer anderen AWS-Konto zugewiesen ist.

Wenn Sie EC2-Classic oder eine Standard-VPC verwenden, wird durch die Freigabe einer Elastic IP-Adresse diese automatisch von allen Instances getrennt, mit der sie verknüpft ist. Verwenden Sie die `disassociateAddress` Methode des EC2 Amazon-Clients, um die Zuordnung einer Elastic IP-Adresse zu trennen, ohne sie freizugeben.

Wenn Sie einen Nicht-Standard-VPC verwenden, müssen Sie die Verknüpfung der Elastic IP-Adresse mit `disassociateAddress` aufheben, bevor Sie versuchen, sie freizugeben. Andernfalls wird ein Fehler Amazon EC2 zurückgegeben (`UngültigIPAddress`). `InUse`).

Siehe [vollständiges Beispiel](#).

## Weitere Informationen

- [Elastische IP-Adressen](#) im Amazon EC2 Benutzerhandbuch für Linux-Instances
- [AllocateAddress](#) in der Amazon EC2 API-Referenz
- [DescribeAddresses](#) in der Amazon EC2 API-Referenz
- [ReleaseAddress](#) in der Amazon EC2 API-Referenz

## Regionen und Verfügbarkeitszonen verwenden

### Beschreiben von Regionen

Rufen Sie die EC2 `describeRegions` Amazon-Client-Methode auf, um die für Ihr Konto verfügbaren Regionen aufzulisten. Sie gibt [DescribeRegionsResult](#) zurück. Rufen Sie die `getRegions`-Methode des zurückgegebenen Objekts auf und Sie erhalten eine Liste mit [Region](#)-Objekten, von denen jedes für eine Region steht.

### Importe

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.DescribeRegionsResult;
import com.amazonaws.services.ec2.model.Region;
import com.amazonaws.services.ec2.model.AvailabilityZone;
import com.amazonaws.services.ec2.model.DescribeAvailabilityZonesResult;
```

### Code

```
DescribeRegionsResult regions_response = ec2.describeRegions();
```

```
for(Region region : regions_response.getRegions()) {
    System.out.printf(
        "Found region %s " +
        "with endpoint %s",
        region.getRegionName(),
        region.getEndpoint());
}
```

Siehe [vollständiges Beispiel](#).

## Beschreiben von Availability Zones

Rufen Sie die EC2 `describeAvailabilityZones` Amazon-Client-Methode auf, um jede Availability Zone aufzulisten, die für Ihr Konto verfügbar sind. Sie gibt [DescribeAvailabilityZonesResult](#) zurück. Rufen Sie die `getAvailabilityZones` Methode auf, um eine Liste von [AvailabilityZone](#) Objekten zu erhalten, die jede Availability Zone repräsentieren.

### Importe

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.DescribeRegionsResult;
import com.amazonaws.services.ec2.model.Region;
import com.amazonaws.services.ec2.model.AvailabilityZone;
import com.amazonaws.services.ec2.model.DescribeAvailabilityZonesResult;
```

### Code

```
DescribeAvailabilityZonesResult zones_response =
    ec2.describeAvailabilityZones();

for(AvailabilityZone zone : zones_response.getAvailabilityZones()) {
    System.out.printf(
        "Found availability zone %s " +
        "with status %s " +
        "in region %s",
        zone.getZoneName(),
        zone.getState(),
        zone.getRegionName());
}
```

Siehe [vollständiges Beispiel](#).

## Beschreiben von Konten

Um Ihr Konto zu beschreiben, rufen Sie die `describeAccountAttributes` Methode des EC2 Amazon-Clients auf. Diese Methode gibt ein [DescribeAccountAttributesResult](#) Objekt zurück. Rufen Sie die `getAccountAttributes` Methode dieses Objekts auf, um eine Liste von [AccountAttribute](#) Objekten zu erhalten. Sie können die Liste durchgehen, um ein [AccountAttribute](#) Objekt abzurufen.

Sie können die Attributwerte Ihres Kontos abrufen, indem Sie die Methode des [AccountAttribute](#) `getAttributeValues` 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 com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.AccountAttributeValue;
import com.amazonaws.services.ec2.model.DescribeAccountAttributesResult;
import com.amazonaws.services.ec2.model.AccountAttribute;
import java.util.List;
import java.util.ListIterator;
```

### Code

```
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

try{
    DescribeAccountAttributesResult accountResults = ec2.describeAccountAttributes();
    List<AccountAttribute> accountList = accountResults.getAccountAttributes();

    for (ListIterator iter = accountList.listIterator(); iter.hasNext(); ) {

        AccountAttribute attribute = (AccountAttribute) iter.next();
        System.out.print("\n The name of the attribute is
"+attribute.getAttributeName());
        List<AccountAttributeValue> values = attribute.getAttributeValues();

        //iterate through the attribute values
        for (ListIterator iterVals = values.listIterator(); iterVals.hasNext(); ) {
```

```
        AccountAttributeValue myValue = (AccountAttributeValue) iterVals.next();
        System.out.print("\n The value of the attribute is
"+myValue.getAttributeValue());
    }
}
System.out.print("Done");
}
catch (Exception e)
{
    e.printStackTrace();
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Weitere Informationen

- [Regionen und Availability Zones](#) im Amazon EC2 Benutzerhandbuch für Linux-Instances
- [DescribeRegions](#) in der Amazon EC2 API-Referenz
- [DescribeAvailabilityZones](#) in der Amazon EC2 API-Referenz

## Mit Amazon EC2 Schlüsselpaaren arbeiten

### Erstellen eines Schlüsselpaars

Um ein key pair zu erstellen, rufen Sie die `createKeyPair` Methode des EC2 Amazon-Clients mit einer auf [CreateKeyPairRequest](#), die den Namen des Schlüssels enthält.

### Importe

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.CreateKeyPairRequest;
import com.amazonaws.services.ec2.model.CreateKeyPairResult;
```

### Code

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

CreateKeyPairRequest request = new CreateKeyPairRequest()
    .withKeyName(key_name);
```

```
CreateKeyPairResult response = ec2.createKeyPair(request);
```

Siehe [vollständiges Beispiel](#).

## Beschreiben von Schlüsselpaaren

Rufen Sie die EC2 `describeKeyPairs` Amazon-Client-Methode auf, um Ihre Schlüsselpaare aufzulisten oder Informationen über sie zu erhalten. Sie gibt eine zurück [DescribeKeyPairsResult](#), mit der Sie auf die Liste der Schlüsselpaare zugreifen können, indem Sie ihre `getKeyPairs` Methode aufrufen, die eine Liste von [KeyPairInfo](#) Objekten zurückgibt.

### Importe

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.DescribeKeyPairsResult;
import com.amazonaws.services.ec2.model.KeyPairInfo;
```

### Code

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

DescribeKeyPairsResult response = ec2.describeKeyPairs();

for(KeyPairInfo key_pair : response.getKeyPairs()) {
    System.out.printf(
        "Found key pair with name %s " +
        "and fingerprint %s",
        key_pair.getKeyName(),
        key_pair.getKeyFingerprint());
}
```

Siehe [vollständiges Beispiel](#).

## Löschen eines Schlüsselpaars

Um ein key pair zu löschen, rufen Sie die `deleteKeyPair` Methode des EC2 Amazon-Clients auf und übergeben Sie ihr eine [DeleteKeyPairRequest](#), die den Namen des zu löschenden key pair enthält.

### Importe

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.DeleteKeyPairRequest;
import com.amazonaws.services.ec2.model.DeleteKeyPairResult;
```

## Code

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

DeleteKeyPairRequest request = new DeleteKeyPairRequest()
    .withKeyName(key_name);

DeleteKeyPairResult response = ec2.deleteKeyPair(request);
```

Siehe [vollständiges Beispiel](#).

## Weitere Informationen

- [Amazon EC2 Schlüsselpaare](#) im Amazon EC2 Benutzerhandbuch für Linux-Instances
- [CreateKeyPair](#) in der Amazon EC2 API-Referenz
- [DescribeKeyPairs](#) in der Amazon EC2 API-Referenz
- [DeleteKeyPair](#) in der Amazon EC2 API-Referenz

# Arbeiten mit Sicherheitsgruppen in Amazon EC2

## Erstellen einer Sicherheitsgruppe

Um eine Sicherheitsgruppe zu erstellen, rufen Sie die `createSecurityGroup` Methode des EC2 Amazon-Clients mit einer auf [CreateSecurityGroupRequest](#), die den Namen des Schlüssels enthält.

## Importe

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.CreateSecurityGroupRequest;
import com.amazonaws.services.ec2.model.CreateSecurityGroupResult;
```

## Code

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

CreateSecurityGroupRequest create_request = new
    CreateSecurityGroupRequest()
        .withGroupName(group_name)
        .withDescription(group_desc)
        .withVpcId(vpc_id);

CreateSecurityGroupResult create_response =
    ec2.createSecurityGroup(create_request);
```

Siehe [vollständiges Beispiel](#).

## Konfigurieren einer Sicherheitsgruppe

Eine Sicherheitsgruppe kann sowohl den eingehenden (eingehenden) als auch den ausgehenden (ausgehenden) Datenverkehr zu Ihren Instances kontrollieren. Amazon EC2

Um Ihrer Sicherheitsgruppe Eingangsregeln hinzuzufügen, verwenden Sie die `authorizeSecurityGroupIngress` Methode des EC2 Amazon-Clients und geben Sie den Namen der Sicherheitsgruppe und die Zugriffsregeln ([IpPermission](#)) an, die Sie ihr innerhalb eines [AuthorizeSecurityGroupIngressRequest](#) Objekts zuweisen möchten. Im folgenden Beispiel wird gezeigt, wie Sie einer Sicherheitsgruppe IP-Berechtigungen hinzufügen.

### Importe

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.CreateSecurityGroupRequest;
import com.amazonaws.services.ec2.model.CreateSecurityGroupResult;
```

### Code

```
IpRange ip_range = new IpRange()
    .withCidrIp("0.0.0.0/0");

IpPermission ip_perm = new IpPermission()
    .withIpProtocol("tcp")
    .withToPort(80)
    .withFromPort(80)
    .withIpv4Ranges(ip_range);
```

```
IpPermission ip_perm2 = new IpPermission()
    .withIpProtocol("tcp")
    .withToPort(22)
    .withFromPort(22)
    .withIpv4Ranges(ip_range);

AuthorizeSecurityGroupIngressRequest auth_request = new
    AuthorizeSecurityGroupIngressRequest()
        .withGroupName(group_name)
        .withIpPermissions(ip_perm, ip_perm2);

AuthorizeSecurityGroupIngressResult auth_response =
    ec2.authorizeSecurityGroupIngress(auth_request);
```

Um der Sicherheitsgruppe eine Ausgangsregel hinzuzufügen, geben Sie ähnliche Daten in einer der `authorizeSecurityGroupEgress` Methoden des EC2 Amazon-Clients [AuthorizeSecurityGroupEgressRequest](#) an.

Siehe [vollständiges Beispiel](#).

## Beschreiben von Sicherheitsgruppen

Rufen Sie die `describeSecurityGroups` Methode des EC2 Amazon-Clients auf, um Ihre Sicherheitsgruppen zu beschreiben oder Informationen über sie zu erhalten. Sie gibt eine zurück [DescribeSecurityGroupsResult](#), mit der Sie auf die Liste der Sicherheitsgruppen zugreifen können, indem Sie ihre `getSecurityGroups` Methode aufrufen, die eine Liste von [SecurityGroup](#) Objekten zurückgibt.

### Importe

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.DescribeSecurityGroupsRequest;
import com.amazonaws.services.ec2.model.DescribeSecurityGroupsResult;
```

### Code

```
final String USAGE =
    "To run this example, supply a group id\n" +
    "Ex: DescribeSecurityGroups <group-id>\n";

if (args.length != 1) {
```

```
        System.out.println(USAGE);
        System.exit(1);
    }

    String group_id = args[0];
```

Siehe [vollständiges Beispiel](#).

## Löschen einer Sicherheitsgruppe

Um eine Sicherheitsgruppe zu löschen, rufen Sie die `deleteSecurityGroup` Methode des EC2 Amazon-Clients auf und übergeben Sie ihr eine [DeleteSecurityGroupRequest](#), die die ID der zu löschenden Sicherheitsgruppe enthält.

### Importe

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.DeleteSecurityGroupRequest;
import com.amazonaws.services.ec2.model.DeleteSecurityGroupResult;
```

### Code

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

DeleteSecurityGroupRequest request = new DeleteSecurityGroupRequest()
    .withGroupId(group_id);

DeleteSecurityGroupResult response = ec2.deleteSecurityGroup(request);
```

Siehe [vollständiges Beispiel](#).

## Weitere Informationen

- [Amazon EC2 Sicherheitsgruppen](#) im Amazon EC2 Benutzerhandbuch für Linux-Instances
- [Autorisieren von eingehendem Traffic für Ihre Linux-Instances](#) im Amazon EC2 Benutzerhandbuch für Linux-Instances
- [CreateSecurityGroup](#) in der API-Referenz Amazon EC2
- [DescribeSecurityGroups](#) in der Amazon EC2 API-Referenz
- [DeleteSecurityGroup](#) in der Amazon EC2 API-Referenz

- [AuthorizeSecurityGroupIngress](#) in der Amazon EC2 API-Referenz

## IAM-Beispiele mit dem AWS SDK für Java

Dieser Abschnitt enthält Beispiele für die Programmierung von [IAM](#) mit dem [AWS SDK für Java](#).

AWS Identity and Access Management (IAM) ermöglicht es Ihnen, den Zugriff Ihrer Benutzer auf AWS Dienste und Ressourcen sicher zu kontrollieren. Mit IAM können Sie AWS Benutzer und Gruppen erstellen und verwalten und ihnen mithilfe von Berechtigungen den Zugriff auf Ressourcen gewähren oder verweigern. AWS Eine vollständige Anleitung zu IAM finden Sie im [IAM Benutzerhandbuch](#).

### Note

Die 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](#)
- [Verwalten von IAM-Benutzern](#)
- [Verwenden von IAM-Konto-Aliasen](#)
- [Arbeiten mit IAM-Richtlinien](#)
- [Arbeiten mit IAM-Serverzertifikaten](#)

## Verwalten von IAM-Zugriffsschlüsseln

### Erstellen eines Zugriffsschlüssels

Um einen IAM-Zugriffsschlüssel zu erstellen, rufen Sie die `AmazonIdentityManagementClient.createAccessKey` Methode mit einem [CreateAccessKeyRequest](#) Objekt auf.

`CreateAccessKeyRequest` hat zwei Konstruktoren — einen für einen Benutzernamen und einen ohne Parameter. Wenn Sie die Version nutzen, die keine Parameter entgegen nimmt, müssen Sie

den Benutzernamen mithilfe der `withUserName`-Setter-Methode festlegen, bevor Sie das Element an die `createAccessKey`-Methode übergeben.

## Importe

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.CreateAccessKeyRequest;
import com.amazonaws.services.identitymanagement.model.CreateAccessKeyResult;
```

## Code

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

CreateAccessKeyRequest request = new CreateAccessKeyRequest()
    .withUserName(user);

CreateAccessKeyResult response = iam.createAccessKey(request);
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Auflisten von Zugriffsschlüsseln

Um die Zugriffsschlüssel für einen bestimmten Benutzer aufzulisten, erstellen Sie ein [ListAccessKeysRequest](#) Objekt, das den Benutzernamen enthält, für den die Schlüssel aufgelistet werden sollen, und übergeben Sie ihn an die `AmazonIdentityManagementClient` `listAccessKeys` Methode.

### Note

Wenn Sie keinen Benutzernamen angeben, wird versucht `listAccessKeys`, die Zugriffsschlüssel aufzulisten, die dem Benutzer zugeordnet sind AWS-Konto, der die Anfrage signiert hat.

## Importe

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
```

```
import com.amazonaws.services.identitymanagement.model.AccessKeyMetadata;
import com.amazonaws.services.identitymanagement.model.ListAccessKeysRequest;
import com.amazonaws.services.identitymanagement.model.ListAccessKeysResult;
```

## Code

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

boolean done = false;
ListAccessKeysRequest request = new ListAccessKeysRequest()
    .withUserName(username);

while (!done) {

    ListAccessKeysResult response = iam.listAccessKeys(request);

    for (AccessKeyMetadata metadata :
        response.getAccessKeyMetadata()) {
        System.out.format("Retrieved access key %s",
            metadata.getAccessKeyId());
    }

    request.setMarker(response.getMarker());

    if (!response.getIsTruncated()) {
        done = true;
    }
}
```

Die Ergebnisse von `listAccessKeys` sind seitenweise angeordnet (mit einem Standardhöchstwert von 100 Datensätzen pro Aufruf). Sie können das zurückgegebene [ListAccessKeysResult](#) Objekt aufrufen `getIsTruncated`, um zu überprüfen, ob die Abfrage weniger Ergebnisse geliefert hat, als verfügbar sind. Falls ja, rufen Sie `setMarker` für den `ListAccessKeysRequest` auf und übergeben Sie ihn beim nächsten Aufruf von `listAccessKeys`.

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Abrufen der letzten Nutzungszeit eines Zugriffsschlüssels

Um die Uhrzeit zu ermitteln, zu der ein Zugriffsschlüssel zuletzt verwendet wurde, rufen Sie die `AmazonIdentityManagementClient` `getAccessKeyLastUsed` 's-Methode mit der ID des

Zugriffsschlüssels auf (die mithilfe eines [GetAccessKeyLastUsedRequest](#) Objekts übergeben werden kann), oder direkt an die Overload, die die Zugriffsschlüssel-ID direkt übernimmt.

Sie können dann das zurückgegebene [GetAccessKeyLastUsedResult](#) Objekt verwenden, um die Uhrzeit der letzten Verwendung des Schlüssels abzurufen.

## Importe

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.GetAccessKeyLastUsedRequest;
import com.amazonaws.services.identitymanagement.model.GetAccessKeyLastUsedResult;
```

## Code

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

GetAccessKeyLastUsedRequest request = new GetAccessKeyLastUsedRequest()
    .withAccessKeyId(access_id);

GetAccessKeyLastUsedResult response = iam.getAccessKeyLastUsed(request);

System.out.println("Access key was last used at: " +
    response.getAccessKeyLastUsed().getLastUsedDate());
```

Das [vollständige Beispiel](#) finden Sie unter 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 den gewünschten [Status](#) angeben und dann das Anforderungsobjekt an die `AmazonIdentityManagementClient` `updateAccessKey` -Methode übergeben.

## Importe

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.UpdateAccessKeyRequest;
```

```
import com.amazonaws.services.identitymanagement.model.UpdateAccessKeyResult;
```

## Code

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

UpdateAccessKeyRequest request = new UpdateAccessKeyRequest()
    .withAccessKeyId(access_id)
    .withUserName(username)
    .withStatus(status);

UpdateAccessKeyResult response = iam.updateAccessKey(request);
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Löschen eines Zugriffsschlüssels

Um einen Zugriffsschlüssel dauerhaft zu löschen, rufen Sie die `deleteKey` Methode `AmazonIdentityManagementClient` auf und geben ihr eine, die die ID und den Benutzernamen des Zugriffsschlüssels [DeleteAccessKeyRequest](#) enthält.

### Note

Nach dem Löschen können Schlüssel nicht mehr abgerufen oder verwendet werden. Um einen Schlüssel vorübergehend zu deaktivieren, sodass er später wieder aktiviert werden kann, verwenden Sie stattdessen [updateAccessKey](#) method.

## Importe

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.DeleteAccessKeyRequest;
import com.amazonaws.services.identitymanagement.model.DeleteAccessKeyResult;
```

## Code

```
final AmazonIdentityManagement iam =
```

```
AmazonIdentityManagementClientBuilder.defaultClient();

DeleteAccessKeyRequest request = new DeleteAccessKeyRequest()
    .withAccessKeyId(access_key)
    .withUserName(username);

DeleteAccessKeyResult response = iam.deleteAccessKey(request);
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Weitere Informationen

- [CreateAccessKey](#) in der IAM-API-Referenz
- [ListAccessKeys](#) in der IAM-API-Referenz
- [GetAccessKeyLastUsed](#) in der IAM-API-Referenz
- [UpdateAccessKey](#) in der IAM-API-Referenz
- [DeleteAccessKey](#) in der IAM-API-Referenz

## Verwalten von IAM-Benutzern

### Erstellen eines Benutzers

Erstellen Sie einen neuen IAM-Benutzer, indem Sie den Benutzernamen für die `createUser` Methode `AmazonIdentityManagementClient` angeben, entweder direkt oder mithilfe eines [CreateUserRequest](#) Objekts, das den Benutzernamen enthält.

### Importe

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.CreateUserRequest;
import com.amazonaws.services.identitymanagement.model.CreateUserResult;
```

### Code

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();
```

```
CreateUserRequest request = new CreateUserRequest()
    .withUserName(username);

CreateUserResult response = iam.createUser(request);
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Auflisten von Benutzern

Um die IAM-Benutzer für Ihr Konto aufzulisten, erstellen Sie ein neues [ListUsersRequest](#) und übergeben Sie es an die `AmazonIdentityManagementClient` `listUsers` Methode. Sie können die Benutzerliste abrufen, indem Sie das zurückgegebene [ListUsersResult](#) Objekt aufrufen `getUsers`.

Die von `listUsers` zurückgegebene Benutzerliste ist segmentiert. Sie können prüfen, ob weitere Ergebnisse bereitliegen, indem Sie die `getIsTruncated`-Methode des Antwortobjekts aufrufen. Gibt sie `true` zurück, rufen Sie die `setMarker()`-Methode des Anfrageobjekts auf und übergeben ihr den Rückgabewert der `getMarker()`-Methode des Antwortobjekts.

### Importe

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.ListUsersRequest;
import com.amazonaws.services.identitymanagement.model.ListUsersResult;
import com.amazonaws.services.identitymanagement.model.User;
```

### Code

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

boolean done = false;
ListUsersRequest request = new ListUsersRequest();

while(!done) {
    ListUsersResult response = iam.listUsers(request);

    for(User user : response.getUsers()) {
        System.out.format("Retrieved user %s", user.getUserName());
    }
}
```

```
request.setMarker(response.getMarker());

if(!response.getIsTruncated()) {
    done = true;
}
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Aktualisieren eines Benutzers

Um einen Benutzer zu aktualisieren, rufen Sie die `updateUser` Methode des `AmazonIdentityManagementClient` Objekts auf, die ein [UpdateUserRequest](#) Objekt verwendet, mit dem Sie den Namen oder Pfad des Benutzers ändern können.

### Importe

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.UpdateUserRequest;
import com.amazonaws.services.identitymanagement.model.UpdateUserResult;
```

### Code

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

UpdateUserRequest request = new UpdateUserRequest()
    .withUserName(cur_name)
    .withNewUserName(new_name);

UpdateUserResult response = iam.updateUser(request);
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Löschen eines Benutzers

Um einen Benutzer zu löschen, rufen Sie die `AmazonIdentityManagementClient` `deleteUser` Anfrage mit einem [UpdateUserRequest](#) Objektsatz mit dem zu löschenden Benutzernamen auf.

### Importe

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.DeleteConflictException;
import com.amazonaws.services.identitymanagement.model.DeleteUserRequest;
```

## Code

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

DeleteUserRequest request = new DeleteUserRequest()
    .withUserName(username);

try {
    iam.deleteUser(request);
} catch (DeleteConflictException e) {
    System.out.println("Unable to delete user. Verify user is not" +
        " associated with any resources");
    throw e;
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Weitere Informationen

- [IAM-Benutzer](#) im IAM Benutzerhandbuch
- [Verwaltung von IAM-Benutzern](#) im Benutzerhandbuch IAM
- [CreateUser](#) in der IAM-API-Referenz
- [ListUsers](#) in der IAM-API-Referenz
- [UpdateUser](#) in der IAM-API-Referenz
- [DeleteUser](#) in der IAM-API-Referenz

## Verwenden von IAM-Konto-Aliasen

Wenn Sie möchten, dass die URL für Ihre Anmeldeseite Ihren Firmennamen oder eine andere benutzerfreundliche Kennung anstelle Ihrer AWS-Konto ID enthält, können Sie einen Alias für Ihre AWS-Konto.

**Note**

AWS unterstützt genau einen Kontoalias pro Konto.

## Erstellen eines Konto-Alias

Um einen Kontoalias zu erstellen, rufen Sie die `createAccountAlias` Methode `AmazonIdentityManagementClient`'s mit einem [CreateAccountAliasRequest](#) Objekt auf, das den Aliasnamen enthält.

### Importe

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.CreateAccountAliasRequest;
import com.amazonaws.services.identitymanagement.model.CreateAccountAliasResult;
```

### Code

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

CreateAccountAliasRequest request = new CreateAccountAliasRequest()
    .withAccountAlias(alias);

CreateAccountAliasResult response = iam.createAccountAlias(request);
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Auflisten von Konto-Aliassen

Rufen Sie die `listAccountAliases` Methode s auf, um den `AmazonIdentityManagementClient` Alias Ihres Kontos aufzulisten, falls vorhanden.

**Note**

Die zurückgegebene Methode [ListAccountAliasesResult](#) unterstützt dieselben `getMarker` Methoden `getIsTruncated` und wie andere AWS SDK für Java Listenmethoden, AWS-Konto kann jedoch nur einen Kontoalias haben.

## Importe

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.ListAccountAliasesResult;
```

## Code

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

ListAccountAliasesResult response = iam.listAccountAliases();

for (String alias : response.getAccountAliases()) {
    System.out.printf("Retrieved account alias %s", alias);
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Löschen eines Konto-Alias

Um den Alias Ihres Kontos zu löschen, rufen Sie `AmazonIdentityManagementClient` die `deleteAccountAlias` Methode auf. Wenn Sie einen Kontoalias löschen, müssen Sie seinen Namen mithilfe eines [DeleteAccountAliasRequest](#) Objekts angeben.

## Importe

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.DeleteAccountAliasRequest;
import com.amazonaws.services.identitymanagement.model.DeleteAccountAliasResult;
```

## Code

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

DeleteAccountAliasRequest request = new DeleteAccountAliasRequest()
    .withAccountAlias(alias);
```

```
DeleteAccountAliasResult response = iam.deleteAccountAlias(request);
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Weitere Informationen

- [Ihre AWS Konto-ID und ihr Alias](#) finden Sie im IAM Benutzerhandbuch
- [CreateAccountAlias](#) in der IAM-API-Referenz
- [ListAccountAliases](#) in der IAM-API-Referenz
- [DeleteAccountAlias](#) in der IAM-API-Referenz

## Arbeiten mit IAM-Richtlinien

### Erstellen einer Richtlinie

Um eine neue Richtlinie zu erstellen, geben Sie den Namen der Richtlinie und ein Richtliniendokument im JSON-Format in einer Methode [CreatePolicyRequest](#) an `AmazonIdentityManagementClient.createPolicy`

### Importe

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;  
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;  
import com.amazonaws.services.identitymanagement.model.CreatePolicyRequest;  
import com.amazonaws.services.identitymanagement.model.CreatePolicyResult;
```

### Code

```
final AmazonIdentityManagement iam =  
    AmazonIdentityManagementClientBuilder.defaultClient();  
  
CreatePolicyRequest request = new CreatePolicyRequest()  
    .withPolicyName(policy_name)  
    .withPolicyDocument(POLICY_DOCUMENT);  
  
CreatePolicyResult response = iam.createPolicy(request);
```

[IAM-Richtliniendokumente sind JSON-Zeichenfolgen mit einer gut dokumentierten Syntax.](#) Hier finden Sie ein Beispiel, das den Zugriff für bestimmte Anfragen an DynamoDB gewährt.

```

public static final String POLICY_DOCUMENT =
    "{" +
    "  \"Version\": \"2012-10-17\",      " +
    "  \"Statement\": [" +
    "    {" +
    "      \"Effect\": \"Allow\", " +
    "      \"Action\": \"logs:CreateLogGroup\", " +
    "      \"Resource\": \"%s\"" +
    "    }, " +
    "    {" +
    "      \"Effect\": \"Allow\", " +
    "      \"Action\": [" +
    "        \"dynamodb:DeleteItem\", " +
    "        \"dynamodb:GetItem\", " +
    "        \"dynamodb:PutItem\", " +
    "        \"dynamodb:Scan\", " +
    "        \"dynamodb:UpdateItem\"" +
    "      ], " +
    "      \"Resource\": \"RESOURCE_ARN\"" +
    "    } " +
    "  ]" +
    "}";

```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Abrufen einer Richtlinie

Um eine bestehende Richtlinie abzurufen, rufen Sie die `AmazonIdentityManagementClient` `getPolicy` Methode auf und geben den ARN der Richtlinie in einem [GetPolicyRequest](#) Objekt an.

### Importe

```

import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.GetPolicyRequest;
import com.amazonaws.services.identitymanagement.model.GetPolicyResult;

```

### Code

```

final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

```

```
GetPolicyRequest request = new GetPolicyRequest()
    .withPolicyArn(policy_arn);

GetPolicyResult response = iam.getPolicy(request);
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Anfügen einer Rollenrichtlinie

Sie können eine Richtlinie an IAM [http://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_roles.html](http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html) [role], indem Sie die `attachRolePolicy` Methode `AmazonIdentityManagementClient`'s aufrufen und ihr den Rollennamen und den Richtlinien-ARN in einem zur Verfügung stellen [AttachRolePolicyRequest](#).

### Importe

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.AttachRolePolicyRequest;
import com.amazonaws.services.identitymanagement.model.AttachedPolicy;
```

### Code

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

AttachRolePolicyRequest attach_request =
    new AttachRolePolicyRequest()
        .withRoleName(role_name)
        .withPolicyArn(POLICY_ARN);

iam.attachRolePolicy(attach_request);
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Auflisten angefügter Rollenrichtlinien

Listet die angehängten Richtlinien für eine Rolle auf, indem Sie die `listAttachedRolePolicies` Methode `AmazonIdentityManagementClient`'s aufrufen. Es wird ein

[ListAttachedRolePoliciesRequest](#) Objekt benötigt, das den Rollennamen enthält, um die Richtlinien aufzulisten.

Rufen Sie das zurückgegebene [ListAttachedRolePoliciesResult](#) Objekt `getAttachedPolicies` auf, um die Liste der angehängten Richtlinien abzurufen. Die Ergebnisse sind evtl. gekürzt. Gibt die `ListAttachedRolePoliciesResult`-Methode des `getIsTruncated`-Objekts `true` zurück, rufen Sie die `ListAttachedRolePoliciesRequest`-Methode des `setMarker`-Objekts auf. Verwenden Sie das Ergebnis dann in einem weiteren Aufruf von `listAttachedRolePolicies`, um das nächste Teilergebnis abzurufen.

## Importe

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.ListAttachedRolePoliciesRequest;
import com.amazonaws.services.identitymanagement.model.ListAttachedRolePoliciesResult;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;
```

## Code

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

ListAttachedRolePoliciesRequest request =
    new ListAttachedRolePoliciesRequest()
        .withRoleName(role_name);

List<AttachedPolicy> matching_policies = new ArrayList<>();

boolean done = false;

while(!done) {
    ListAttachedRolePoliciesResult response =
        iam.listAttachedRolePolicies(request);

    matching_policies.addAll(
        response.getAttachedPolicies()
            .stream()
            .filter(p -> p.getPolicyName().equals(role_name))
            .collect(Collectors.toList()));
}
```

```
if(!response.getIsTruncated()) {
    done = true;
}
request.setMarker(response.getMarker());
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Trennen einer Rollenrichtlinie

Um eine Richtlinie von einer Rolle zu trennen, rufen Sie die `detachRolePolicy` Methode `AmazonIdentityManagementClient`'s auf und geben Sie ihr den Rollennamen und den Richtlinien-ARN in a [DetachRolePolicyRequest](#).

### Importe

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.DetachRolePolicyRequest;
import com.amazonaws.services.identitymanagement.model.DetachRolePolicyResult;
```

### Code

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

DetachRolePolicyRequest request = new DetachRolePolicyRequest()
    .withRoleName(role_name)
    .withPolicyArn(policy_arn);

DetachRolePolicyResult response = iam.detachRolePolicy(request);
```

Das [vollständige Beispiel](#) finden Sie unter. GitHub

## Weitere Informationen

- [Überblick über die IAM-Richtlinien](#) im IAM Benutzerhandbuch.
- [AWS Referenz zu den IAM-Richtlinien](#) im IAM Benutzerhandbuch.
- [CreatePolicy](#) in der IAM-API-Referenz

- [GetPolicy](#) in der IAM-API-Referenz
- [AttachRolePolicy](#) in der IAM-API-Referenz
- [ListAttachedRolePolicies](#) in der IAM-API-Referenz
- [DetachRolePolicy](#) in der IAM-API-Referenz

## Arbeiten mit IAM-Serverzertifikaten

Um HTTPS-Verbindungen zu Ihrer Website oder Anwendung zu aktivieren AWS, benötigen Sie ein SSL/TLS-Serverzertifikat. Sie können ein Serverzertifikat verwenden, das von AWS Certificate Manager bereitgestellt wird, oder eines, das Sie von einem externen Anbieter bezogen haben.

Wir empfehlen, dass Sie ACM für die Bereitstellung, Verwaltung und Bereitstellung Ihrer Serverzertifikate verwenden. Mit ACM können Sie ein Zertifikat anfordern, es für Ihre AWS Ressourcen bereitstellen und ACM die Zertifikatserneuerung für Sie durchführen lassen. Zertifikate von ACM sind kostenlos. [Weitere Informationen zu ACM finden Sie im ACM-Benutzerhandbuch.](#)

### Abrufen eines Serverzertifikats

Sie können ein Serverzertifikat abrufen, indem Sie die `getServerCertificate` Methode `AmazonIdentityManagementClient` aufrufen und ihr eine [GetServerCertificateRequest](#) mit dem Namen des Zertifikats übergeben.

### Importe

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.GetServerCertificateRequest;
import com.amazonaws.services.identitymanagement.model.GetServerCertificateResult;
```

### Code

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

GetServerCertificateRequest request = new GetServerCertificateRequest()
    .withServerCertificateName(cert_name);

GetServerCertificateResult response = iam.getServerCertificate(request);
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Auflisten von Serverzertifikaten

Um Ihre Serverzertifikate aufzulisten, rufen Sie die `listServerCertificates` Methode `AmazonIdentityManagementClient`'s mit einem auf [ListServerCertificatesRequest](#). Sie gibt [ListServerCertificatesResult](#) zurück.

Rufen Sie die `getServerCertificateMetadataList` Methode des zurückgegebenen `ListServerCertificateResult` Objekts auf, um eine Liste von [ServerCertificateMetadata](#) Objekten abzurufen, mit denen Sie Informationen zu den einzelnen Zertifikaten abrufen können.

Die Ergebnisse sind evtl. gekürzt. Gibt die `ListServerCertificateResult`-Methode des `getIsTruncated`-Objekts `true` zurück, rufen Sie die `ListServerCertificatesRequest`-Methode des `setMarker`-Objekts auf. Verwenden Sie das Ergebnis dann in einem weiteren Aufruf von `listServerCertificates`, um das nächste Teilergebnis abzurufen.

### Importe

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.ListServerCertificatesRequest;
import com.amazonaws.services.identitymanagement.model.ListServerCertificatesResult;
import com.amazonaws.services.identitymanagement.model.ServerCertificateMetadata;
```

### Code

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

boolean done = false;
ListServerCertificatesRequest request =
    new ListServerCertificatesRequest();

while(!done) {

    ListServerCertificatesResult response =
        iam.listServerCertificates(request);

    for(ServerCertificateMetadata metadata :
```

```
        response.getServerCertificateMetadataList() {
            System.out.printf("Retrieved server certificate %s",
                metadata.getServerCertificateName());
        }

        request.setMarker(response.getMarker());

        if(!response.getIsTruncated()) {
            done = true;
        }
    }
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Aktualisieren eines Serverzertifikats

Sie können den Namen oder Pfad eines Serverzertifikats aktualisieren, indem Sie die `updateServerCertificate` Methode `AmazonIdentityManagementClient`'s aufrufen. Sie benötigt eine [UpdateServerCertificateRequest](#) Objektgruppe mit dem aktuellen Namen des Serverzertifikats und entweder einem neuen Namen oder einem neuen Pfad zur Verwendung.

### Importe

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.UpdateServerCertificateRequest;
import com.amazonaws.services.identitymanagement.model.UpdateServerCertificateResult;
```

### Code

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

UpdateServerCertificateRequest request =
    new UpdateServerCertificateRequest()
        .withServerCertificateName(cur_name)
        .withNewServerCertificateName(new_name);

UpdateServerCertificateResult response =
    iam.updateServerCertificate(request);
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Löschen eines Serverzertifikats

Um ein Serverzertifikat zu löschen, rufen Sie die `deleteServerCertificate` Methode `AmazonIdentityManagementClient` mit einem auf, das den Namen des Zertifikats [DeleteServerCertificateRequest](#) enthält.

### Importe

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.DeleteServerCertificateRequest;
import com.amazonaws.services.identitymanagement.model.DeleteServerCertificateResult;
```

### Code

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

DeleteServerCertificateRequest request =
    new DeleteServerCertificateRequest()
        .withServerCertificateName(cert_name);

DeleteServerCertificateResult response =
    iam.deleteServerCertificate(request);
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

### Weitere Informationen

- [Arbeiten mit Serverzertifikaten](#) im IAM Benutzerhandbuch
- [GetServerCertificate](#) in der IAM-API-Referenz
- [ListServerCertificates](#) in der IAM-API-Referenz
- [UpdateServerCertificate](#) in der IAM-API-Referenz
- [DeleteServerCertificate](#) in der IAM-API-Referenz
- [ACM-Benutzerhandbuch](#)

## Lambda Beispiele für die Verwendung der AWS SDK für Java

Dieser Abschnitt enthält Beispiele für die Programmierung Lambda mit dem AWS SDK für Java.

**Note**

Die 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

- [Funktionen aufrufen, auflisten und löschen Lambda](#)

## Funktionen aufrufen, auflisten und löschen Lambda

Dieser Abschnitt enthält Beispiele für die Programmierung mit dem Lambda Service-Client unter Verwendung von. AWS SDK für Java Informationen zum Erstellen einer Lambda Funktion finden Sie unter [So erstellen Sie AWS Lambda Funktionen](#).

## Themen

- [Aufruf einer -Funktion](#)
- [Listenfunktionen](#)
- [Löschen einer -Funktion](#)

### Aufruf einer -Funktion

Sie können eine Lambda Funktion aufrufen, indem Sie ein [AWSLambda](#)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 Funktion übergeben werden sollen. Lambda Funktionsnamen werden als `arn:aws:lambda:us-east - 1:555556330391:function: angezeigt. HelloFunction` Sie können den Wert abrufen, indem Sie sich die Funktion in der ansehen. AWS-Managementkonsole

Um Nutzdaten an eine Funktion zu übergeben, rufen Sie die `withPayload` Methode des [InvokeRequest](#)Objekts auf und geben Sie eine Zeichenfolge im JSON-Format an, wie im folgenden Codebeispiel gezeigt.

## Importe

```
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.lambda.AWSLambda;
import com.amazonaws.services.lambda.AWSLambdaClientBuilder;
import com.amazonaws.services.lambda.model.InvokeRequest;
import com.amazonaws.services.lambda.model.InvokeResult;
import com.amazonaws.services.lambda.model.ServiceException;

import java.nio.charset.StandardCharsets;
```

## Code

Das folgende Codebeispiel zeigt, wie eine Funktion aufgerufen wird. Lambda

```
String functionName = args[0];

InvokeRequest invokeRequest = new InvokeRequest()
    .withFunctionName(functionName)
    .withPayload("{\n" +
        "  \"Hello \": \"Paris\",\n" +
        "  \"countryCode\": \"FR\"\n" +
        "}");
InvokeResult invokeResult = null;

try {
    AWSLambda awsLambda = AWSLambdaClientBuilder.standard()
        .withCredentials(new ProfileCredentialsProvider())
        .withRegion(Regions.US_WEST_2).build();

    invokeResult = awsLambda.invoke(invokeRequest);

    String ans = new String(invokeResult.getPayload().array(),
        StandardCharsets.UTF_8);

    //write out the return value
    System.out.println(ans);

} catch (ServiceException e) {
    System.out.println(e);
}

System.out.println(invokeResult.getStatusCode());
```

Auf GitHub finden Sie ein [vollständiges Beispiel](#).

## Listenfunktionen

Erstellen Sie ein [AWSLambda](#) Objekt und rufen Sie seine `listFunctions` Methode auf. Diese Methode gibt ein [ListFunctionsResult](#) Objekt zurück. Sie können die `getFunctions` 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 com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.lambda.AWSLambda;
import com.amazonaws.services.lambda.AWSLambdaClientBuilder;
import com.amazonaws.services.lambda.model.FunctionConfiguration;
import com.amazonaws.services.lambda.model.ListFunctionsResult;
import com.amazonaws.services.lambda.model.ServiceException;
import java.util.Iterator;
import java.util.List;
```

### Code

Das folgende Java-Codebeispiel zeigt, wie eine Liste von Lambda Funktionsnamen abgerufen wird.

```
ListFunctionsResult functionResult = null;

try {
    AWSLambda awsLambda = AWSLambdaClientBuilder.standard()
        .withCredentials(new ProfileCredentialsProvider())
        .withRegion(Regions.US_WEST_2).build();

    functionResult = awsLambda.listFunctions();

    List<FunctionConfiguration> list = functionResult.getFunctions();

    for (Iterator iter = list.iterator(); iter.hasNext(); ) {
        FunctionConfiguration config = (FunctionConfiguration)iter.next();

        System.out.println("The function name is "+config.getFunctionName());
    }
}
```

```
    } catch (ServiceException e) {  
        System.out.println(e);  
    }  
}
```

Auf GitHub finden Sie ein [vollständiges Beispiel](#).

## Löschen einer -Funktion

Erstellen Sie ein [AWSLambda](#) Objekt und rufen Sie seine `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 - 1:555556330391:function:` angezeigt. HelloFunction Sie können den Wert abrufen, indem Sie sich die Funktion in der ansehen. AWS-Managementkonsole

### Importe

```
import com.amazonaws.auth.profile.ProfileCredentialsProvider;  
import com.amazonaws.regions.Regions;  
import com.amazonaws.services.lambda.AWSLambda;  
import com.amazonaws.services.lambda.AWSLambdaClientBuilder;  
import com.amazonaws.services.lambda.model.ServiceException;  
import com.amazonaws.services.lambda.model.DeleteFunctionRequest;
```

### Code

Der folgende Java-Code zeigt, wie eine Lambda Funktion gelöscht wird.

```
String functionName = args[0];  
try {  
    AWSLambda awsLambda = AWSLambdaClientBuilder.standard()  
        .withCredentials(new ProfileCredentialsProvider())  
        .withRegion(Regions.US_WEST_2).build();  
  
    DeleteFunctionRequest delFunc = new DeleteFunctionRequest();  
    delFunc.withFunctionName(functionName);  
  
    //Delete the function  
    awsLambda.deleteFunction(delFunc);  
    System.out.println("The function is deleted");  
  
} catch (ServiceException e) {  
    System.out.println(e);  
}
```

```
}
```

Auf GitHub finden Sie ein [vollständiges Beispiel](#).

## Amazon Pinpoint Beispiele für die Verwendung der AWS SDK für Java

Dieser Abschnitt bietet Beispiele für die Programmierung von [Amazon Pinpoint](#) mithilfe des [AWS SDK für Java](#).

### Note

Die 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

- [Apps erstellen und löschen in Amazon Pinpoint](#)
- [Endpunkte erstellen in Amazon Pinpoint](#)
- [Segmente erstellen in Amazon Pinpoint](#)
- [Kampagnen erstellen in Amazon Pinpoint](#)
- [Kanäle aktualisieren in Amazon Pinpoint](#)

## Apps erstellen und löschen in Amazon Pinpoint

Eine App ist ein Amazon Pinpoint Projekt, in dem Sie die Zielgruppe für eine bestimmte Anwendung definieren und diese Zielgruppe mit maßgeschneiderten Nachrichten ansprechen. Die Beispiele auf dieser Seite zeigen, wie Sie eine neue App erstellen oder eine bestehende löschen.

### Erstellen einer Anwendung

Erstellen Sie eine neue App, Amazon Pinpoint indem Sie dem [CreateAppRequest](#) Objekt einen App-Namen geben und dieses Objekt dann an die `AmazonPinpointClient createApp` -Methode übergeben.

## Importe

```
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.CreateAppRequest;
import com.amazonaws.services.pinpoint.model.CreateAppResult;
import com.amazonaws.services.pinpoint.model.CreateApplicationRequest;
```

## Code

```
CreateApplicationRequest appRequest = new CreateApplicationRequest()
    .withName(appName);

CreateAppRequest request = new CreateAppRequest();
request.withCreateApplicationRequest(appRequest);
CreateAppResult result = pinpoint.createApp(request);
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Löschen einer APP

Um eine App zu löschen, rufen Sie die AmazonPinpointClient deleteApp Anfrage mit einem [DeleteAppRequest](#) Objekt auf, für das der zu löschende App-Name festgelegt ist.

## Importe

```
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
```

## Code

```
DeleteAppRequest deleteRequest = new DeleteAppRequest()
    .withApplicationId(appID);

pinpoint.deleteApp(deleteRequest);
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Weitere Informationen

- [Apps](#) in der Amazon Pinpoint API-Referenz

- [App](#) in der Amazon Pinpoint API-Referenz

## Endpunkte erstellen in Amazon Pinpoint

Ein Endpunkt kennzeichnet auf eindeutige Weise ein Benutzergerät, an das Sie mit Amazon Pinpoint Push-Benachrichtigungen senden können. Wenn für Ihre App Amazon Pinpoint Support aktiviert ist, registriert Ihre App automatisch einen Endpunkt, Amazon Pinpoint wenn ein neuer Benutzer Ihre App öffnet. Das folgende Beispiel zeigt, wie Sie programmgesteuert einen neuen Endpunkt hinzufügen.

### Erstellen eines Endpunkts

Erstellen Sie einen neuen Endpunkt, Amazon Pinpoint indem Sie die Endpunktdaten in einem [EndpointRequest](#) Objekt angeben.

#### Importe

```
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.UpdateEndpointRequest;
import com.amazonaws.services.pinpoint.model.UpdateEndpointResult;
import com.amazonaws.services.pinpoint.model.EndpointDemographic;
import com.amazonaws.services.pinpoint.model.EndpointLocation;
import com.amazonaws.services.pinpoint.model.EndpointRequest;
import com.amazonaws.services.pinpoint.model.EndpointResponse;
import com.amazonaws.services.pinpoint.model.EndpointUser;
import com.amazonaws.services.pinpoint.model.GetEndpointRequest;
import com.amazonaws.services.pinpoint.model.GetEndpointResult;
```

#### Code

```
HashMap<String, List<String>> customAttributes = new HashMap<>();
List<String> favoriteTeams = new ArrayList<>();
favoriteTeams.add("Lakers");
favoriteTeams.add("Warriors");
customAttributes.put("team", favoriteTeams);

EndpointDemographic demographic = new EndpointDemographic()
    .withAppVersion("1.0")
    .withMake("apple")
    .withModel("iPhone")
    .withModelVersion("7")
```

```
.withPlatform("ios")
.withPlatformVersion("10.1.1")
.withTimezone("America/Los_Angeles");

EndpointLocation location = new EndpointLocation()
    .withCity("Los Angeles")
    .withCountry("US")
    .withLatitude(34.0)
    .withLongitude(-118.2)
    .withPostalCode("90068")
    .withRegion("CA");

Map<String,Double> metrics = new HashMap<>();
metrics.put("health", 100.00);
metrics.put("luck", 75.00);

EndpointUser user = new EndpointUser()
    .withUserId(UUID.randomUUID().toString());

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());

EndpointRequest endpointRequest = new EndpointRequest()
    .withAddress(UUID.randomUUID().toString())
    .withAttributes(customAttributes)
    .withChannelType("APNS")
    .withDemographic(demographic)
    .withEffectiveDate(nowAsISO)
    .withLocation(location)
    .withMetrics(metrics)
    .withOptOut("NONE")
    .withRequestId(UUID.randomUUID().toString())
    .withUser(user);
```

Erstellen Sie dann ein [UpdateEndpointRequest](#) Objekt mit diesem EndpointRequest Objekt. Schließlich übergeben Sie das UpdateEndpointRequest Objekt an AmazonPinpointClient die updateEndpoint Methode.

## Code

```
UpdateEndpointRequest updateEndpointRequest = new UpdateEndpointRequest()
    .withApplicationId(appId)
```

```
        .withEndpointId(endpointId)
        .withEndpointRequest(endpointRequest);

UpdateEndpointResult updateEndpointResponse =
    client.updateEndpoint(updateEndpointRequest);
System.out.println("Update Endpoint Response: " +
    updateEndpointResponse.getMessageBody());
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Weitere Informationen

- [Hinzufügen von Endpunkten](#) im Amazon Pinpoint Entwicklerhandbuch
- [Endpunkt](#) in der Amazon Pinpoint API-Referenz

## Segmente erstellen in Amazon Pinpoint

Ein Benutzersegment stellt einen Teil Ihrer Benutzer basierend auf gemeinsamen Merkmalen dar, z. B. Zeitpunkt der letzten App-Öffnung durch einen Benutzer oder verwendetes Gerät. Das folgende Beispiel zeigt, wie ein Benutzersegment definiert wird.

### Erstellen eines Segments

Erstellen Sie ein neues Segment in, Amazon Pinpoint indem Sie die Abmessungen des Segments in einem [SegmentDimensions](#) Objekt definieren.

#### Importe

```
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.CreateSegmentRequest;
import com.amazonaws.services.pinpoint.model.CreateSegmentResult;
import com.amazonaws.services.pinpoint.model.AttributeDimension;
import com.amazonaws.services.pinpoint.model.AttributeType;
import com.amazonaws.services.pinpoint.model.RecencyDimension;
import com.amazonaws.services.pinpoint.model.SegmentBehaviors;
import com.amazonaws.services.pinpoint.model.SegmentDemographics;
import com.amazonaws.services.pinpoint.model.SegmentDimensions;
import com.amazonaws.services.pinpoint.model.SegmentLocation;
import com.amazonaws.services.pinpoint.model.SegmentResponse;
import com.amazonaws.services.pinpoint.model.WriteSegmentRequest;
```

## Code

```
Pinpoint pinpoint =
    AmazonPinpointClientBuilder.standard().withRegion(Regions.US_EAST_1).build();
Map<String, AttributeDimension> segmentAttributes = new HashMap<>();
segmentAttributes.put("Team", new
    AttributeDimension().withAttributeType(AttributeType.INCLUSIVE).withValues("Lakers"));

SegmentBehaviors segmentBehaviors = new SegmentBehaviors();
SegmentDemographics segmentDemographics = new SegmentDemographics();
SegmentLocation segmentLocation = new SegmentLocation();

RecencyDimension recencyDimension = new RecencyDimension();
recencyDimension.withDuration("DAY_30").withRecencyType("ACTIVE");
segmentBehaviors.setRecency(recencyDimension);

SegmentDimensions dimensions = new SegmentDimensions()
    .withAttributes(segmentAttributes)
    .withBehavior(segmentBehaviors)
    .withDemographic(segmentDemographics)
    .withLocation(segmentLocation);
```

Als Nächstes setzen Sie das [SegmentDimensions](#) Objekt in ein [WriteSegmentRequest](#), das wiederum verwendet wird, um ein [CreateSegmentRequest](#) Objekt zu erstellen. Übergeben Sie dann das [CreateSegmentRequest](#) Objekt an [AmazonPinpointClient](#) die `createSegment` Methode.

## Code

```
WriteSegmentRequest writeSegmentRequest = new WriteSegmentRequest()
    .withName("MySegment").withDimensions(dimensions);

CreateSegmentRequest createSegmentRequest = new CreateSegmentRequest()
    .withApplicationId(appId).withWriteSegmentRequest(writeSegmentRequest);

CreateSegmentResult createSegmentResult = client.createSegment(createSegmentRequest);
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#).

## Weitere Informationen

- [Amazon Pinpoint Segmente](#) im Amazon Pinpoint Benutzerhandbuch
- [Segmente im Amazon Pinpoint Entwicklerhandbuch erstellen](#)

- [Segmente](#) in der Amazon Pinpoint API-Referenz
- [Segment](#) in der Amazon Pinpoint API-Referenz

## Kampagnen erstellen in Amazon Pinpoint

Mit diesen Kampagnen können Sie die Bindung zwischen Ihrer App und den Benutzern erhöhen. Sie können eine Kampagne erstellen, um für ein bestimmtes Benutzersegment maßgeschneiderte Nachrichten oder besondere Werbeaktionen bereitzustellen. In diesem Beispiel wird gezeigt, wie eine neue Standard-Kampagne erstellt wird, bei der eine benutzerdefinierte Push-Benachrichtigung an ein bestimmtes Benutzersegment gesendet wird.

### Erstellen einer Kampagne

Bevor Sie eine neue Kampagne erstellen, müssen Sie einen [Zeitplan](#) und eine [Nachricht](#) definieren und diese Werte in einem [WriteCampaignRequest](#) Objekt festlegen.

#### Importe

```
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.CreateCampaignRequest;
import com.amazonaws.services.pinpoint.model.CreateCampaignResult;
import com.amazonaws.services.pinpoint.model.Action;
import com.amazonaws.services.pinpoint.model.CampaignResponse;
import com.amazonaws.services.pinpoint.model.Message;
import com.amazonaws.services.pinpoint.model.MessageConfiguration;
import com.amazonaws.services.pinpoint.model.Schedule;
import com.amazonaws.services.pinpoint.model.WriteCampaignRequest;
```

#### Code

```
Schedule schedule = new Schedule()
    .withStartTime("IMMEDIATE");

Message defaultMessage = new Message()
    .withAction(Action.OPEN_APP)
    .withBody("My message body.")
    .withTitle("My message title.");

MessageConfiguration messageConfiguration = new MessageConfiguration()
```

```
        .withDefaultMessage(defaultMessage);

WriteCampaignRequest request = new WriteCampaignRequest()
    .withDescription("My description.")
    .withSchedule(schedule)
    .withSegmentId(segmentId)
    .withName("MyCampaign")
    .withMessageConfiguration(messageConfiguration);
```

Erstellen Sie dann eine neue Kampagne, Amazon Pinpoint indem Sie die [WriteCampaignRequest](#) Kampagnenkonfiguration für ein [CreateCampaignRequest](#) Objekt angeben. Schließlich übergeben Sie das CreateCampaignRequest Objekt an AmazonPinpointClient die createCampaign Methode.

## Code

```
CreateCampaignRequest createCampaignRequest = new CreateCampaignRequest()
    .withApplicationId(appId).withWriteCampaignRequest(request);

CreateCampaignResult result = client.createCampaign(createCampaignRequest);
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Weitere Informationen

- [Amazon Pinpoint Kampagnen](#) im Amazon Pinpoint Benutzerhandbuch
- [Kampagnen im Amazon Pinpoint Entwicklerhandbuch erstellen](#)
- [Kampagnen](#) in der Amazon Pinpoint API-Referenz
- [Kampagne](#) in der Amazon Pinpoint API-Referenz
- [Kampagnenaktivitäten](#) in der Amazon Pinpoint API-Referenz
- [Kampagnenversionen](#) in der Amazon Pinpoint API-Referenz
- [Kampagnenversion](#) in der Amazon Pinpoint API-Referenz

## Kanäle aktualisieren in Amazon Pinpoint

Ein Channel definiert die Arten von Plattformen, an die Sie Nachrichten übermitteln können. Dieses Beispiel zeigt, wie der APNs Kanal zum Senden einer Nachricht verwendet wird.

## Aktualisieren eines Channels

Aktivieren Sie einen Kanal, Amazon Pinpoint indem Sie eine App-ID und ein Anforderungsobjekt des Kanaltyps angeben, den Sie aktualisieren möchten. In diesem Beispiel wird der APNs Kanal aktualisiert, wofür das [APNSChannelRequest-Objekt](#) erforderlich ist. Legen Sie diese in der Methode fest [UpdateApnsChannelRequest](#) und übergeben Sie das Objekt an AmazonPinpointClient die `updateApnsChannel` Methode.

### Importe

```
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.APNSChannelRequest;
import com.amazonaws.services.pinpoint.model.APNSChannelResponse;
import com.amazonaws.services.pinpoint.model.GetApnsChannelRequest;
import com.amazonaws.services.pinpoint.model.GetApnsChannelResult;
import com.amazonaws.services.pinpoint.model.UpdateApnsChannelRequest;
import com.amazonaws.services.pinpoint.model.UpdateApnsChannelResult;
```

### Code

```
APNSChannelRequest request = new APNSChannelRequest()
    .withEnabled(enabled);

UpdateApnsChannelRequest updateRequest = new UpdateApnsChannelRequest()
    .withAPNSChannelRequest(request)
    .withApplicationId(appId);
UpdateApnsChannelResult result = client.updateApnsChannel(updateRequest);
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Weitere Informationen

- [Amazon Pinpoint Kanäle](#) im Amazon Pinpoint Benutzerhandbuch
- [ADM-Kanal](#) in der Amazon Pinpoint API-Referenz
- [APNs Kanal](#) in der Amazon Pinpoint API-Referenz
- [APNs Sandbox-Kanal](#) in der Amazon Pinpoint API-Referenz
- [APNs VoIP-Kanal](#) in der Amazon Pinpoint API-Referenz
- [APNs VoIP-Sandbox-Kanal](#) in der Amazon Pinpoint API-Referenz

- Der [Baidu-Kanal](#) in der API-Referenz Amazon Pinpoint
- [E-Mail-Kanal](#) in der Amazon Pinpoint API-Referenz
- [GCM-Kanal](#) in der Amazon Pinpoint API-Referenz
- [SMS-Kanal](#) in der Amazon Pinpoint API-Referenz

## Amazon S3 Beispiele für die Verwendung der AWS SDK für Java

Dieser Abschnitt bietet Beispiele für die Programmierung von [Amazon S3](#) mithilfe des [AWS SDK für Java](#).

### Note

Die 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

- [Amazon S3 Buckets erstellen, auflisten und löschen](#)
- [Operationen an Amazon S3 Objekten ausführen](#)
- [Amazon S3 Zugriffsberechtigungen für Buckets und Objekte verwalten](#)
- [Verwaltung des Zugriffs auf Amazon S3 Buckets mithilfe von Bucket-Richtlinien](#)
- [TransferManager Für Amazon S3 Operationen verwenden](#)
- [Einen Amazon S3 Bucket als Website konfigurieren](#)
- [Amazon S3 Clientseitige Verschlüsselung verwenden](#)

## Amazon S3 Buckets erstellen, auflisten und löschen

Jedes Objekt (Datei) Amazon S3 muss sich in einem Bucket befinden, der eine Sammlung (Container) von Objekten darstellt. Jeder Bucket ist mit einem Schlüssel (Namen) bekannt, der eindeutig sein muss. Ausführliche Informationen zu Buckets und ihrer Konfiguration finden Sie unter [Arbeiten mit 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 darauf Amazon S3 hin, dass mehrteilige Uploads abgebrochen werden, die nicht innerhalb einer bestimmten Anzahl von Tagen nach der Initiierung abgeschlossen werden. Wenn das festgelegte Zeitlimit überschritten wird, wird der Upload Amazon S3 abgebrochen und anschließend die unvollständigen Upload-Daten gelöscht.

Weitere Informationen finden Sie unter [Lebenszykluskonfiguration für einen Bucket mit Versionierung](#) im Amazon S3 Benutzerhandbuch.

**Note**

Bei diesen Codebeispielen wird vorausgesetzt, dass Sie die Informationen [unter Verwenden von](#) verstehen AWS SDK für Java und AWS Standardanmeldedaten anhand der Informationen unter [AWS Anmeldeinformationen einrichten und Region für die Entwicklung](#) konfiguriert haben.

## Bucket erstellen

Verwenden Sie die Methode des AmazonS3-Clients. `createBucket` Der neue [Bucket](#) wird zurückgegeben. Die `createBucket`-Methode löst eine Ausnahme aus, falls der Bucket bereits vorhanden ist.

**Note**

Bevor Sie versuchen, einen Bucket zu erstellen, sollten Sie die `doesBucketExist`-Methode aufrufen, um zu prüfen, ob ein gleichnamiger Bucket bereits existiert. Falls ja, wird `true` zurückgegeben, andernfalls `false`.

## Importe

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
```

```
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AmazonS3Exception;
import com.amazonaws.services.s3.model.Bucket;

import java.util.List;
```

## Code

```
if (s3.doesBucketExistV2(bucket_name)) {
    System.out.format("Bucket %s already exists.\n", bucket_name);
    b = getBucket(bucket_name);
} else {
    try {
        b = s3.createBucket(bucket_name);
    } catch (AmazonS3Exception e) {
        System.err.println(e.getMessage());
    }
}
return b;
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Auflisten von Buckets

Verwenden Sie die Methode des AmazonS3-Clients. `listBucket` Wenn diese Aktion erfolgreich ist, wird eine Liste mit [Bucket](#)-Objekten zurückgegeben.

## Importe

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.Bucket;

import java.util.List;
```

## Code

```
List<Bucket> buckets = s3.listBuckets();
System.out.println("Your {S3} buckets are:");
for (Bucket b : buckets) {
```

```
System.out.println("* " + b.getName());
}
```

Das [vollständige Beispiel](#) finden Sie unter. GitHub

## Bucket löschen

Bevor Sie einen Amazon S3 Bucket löschen können, müssen Sie sicherstellen, dass der Bucket leer ist. Andernfalls tritt ein Fehler auf. Wenn Sie einen [versionierten Bucket](#) nutzen, müssen Sie außerdem alle versionierten Objekte löschen, die mit dem Bucket verknüpft sind.

### Note

Das [vollständige Beispiel](#) umfasst die einzelnen Schritte der Reihe nach und bietet somit eine vollständige Lösung für das Löschen eines Amazon S3 Buckets und seines Inhalts.

## Themen

- [Entfernen von Objekten aus einem nicht versionierten Bucket vor dem Löschen](#)
- [Entfernen von Objekten aus einem versionierten Bucket vor dem Löschen](#)
- [Löschen eines leeren Buckets](#)

## Entfernen von Objekten aus einem nicht versionierten Bucket vor dem Löschen

Verwenden Sie die `listObjects` Methode des AmazonS3-Clients, um die Objektliste abzurufen und jedes Objekt `deleteObject` zu löschen.

## Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.Iterator;
```

## Code

```
System.out.println(" - removing objects from bucket");
ObjectListing object_listing = s3.listObjects(bucket_name);
while (true) {
    for (Iterator<?> iterator =
        object_listing.getObjectSummaries().iterator();
        iterator.hasNext(); ) {
        S3ObjectSummary summary = (S3ObjectSummary) iterator.next();
        s3.deleteObject(bucket_name, summary.getKey());
    }

    // more object_listing to retrieve?
    if (object_listing.isTruncated()) {
        object_listing = s3.listNextBatchOfObjects(object_listing);
    } else {
        break;
    }
}
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

### Entfernen von Objekten aus einem versionierten Bucket vor dem Löschen

Wenn Sie einen [versionierten Bucket](#) nutzen, müssen Sie auch alle gespeicherten Versionen der Objekte im Bucket entfernen, bevor der Bucket gelöscht werden kann.

Verwenden Sie ein ähnliches Muster wie beim Entfernen von Objekten innerhalb eines Buckets. Entfernen Sie versionierte Objekte, indem Sie die `listVersions` Methode des AmazonS3-Clients verwenden, um alle versionierten Objekte aufzulisten und dann jedes einzelne `deleteVersion` zu löschen.

### Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.Iterator;
```

### Code

```
System.out.println(" - removing versions from bucket");
VersionListing version_listing = s3.listVersions(
    new ListVersionsRequest().withBucketName(bucket_name));
while (true) {
    for (Iterator<?> iterator =
        version_listing.getVersionSummaries().iterator();
        iterator.hasNext(); ) {
        S3VersionSummary vs = (S3VersionSummary) iterator.next();
        s3.deleteVersion(
            bucket_name, vs.getKey(), vs.getVersionId());
    }

    if (version_listing.isTruncated()) {
        version_listing = s3.listNextBatchOfVersions(
            version_listing);
    } else {
        break;
    }
}
```

[Das vollständige Beispiel finden Sie unter. GitHub](#)

## Löschen eines leeren Buckets

Sobald Sie die Objekte aus einem Bucket entfernt haben (einschließlich aller versionierten Objekte), können Sie den Bucket selbst mithilfe der Methode des AmazonS3-Clients löschen. `deleteBucket`

## Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.Iterator;
```

## Code

```
System.out.println(" OK, bucket ready to delete!");
s3.deleteBucket(bucket_name);
```

Das [vollständige](#) Beispiel finden Sie unter [GitHub](#)

## Operationen an Amazon S3 Objekten ausführen

Ein Amazon S3 Objekt steht für eine Datei oder eine Sammlung von Daten. Jedes Objekt muss in einem [Bucket](#) enthalten sein.

### Note

Bei diesen Codebeispielen wird vorausgesetzt, dass Sie die Informationen [unter Verwenden von](#) verstehen AWS SDK für Java und AWS Standardanmeldedaten anhand der Informationen unter [AWS Anmeldeinformationen einrichten und Region für die Entwicklung](#) konfiguriert haben.

### Themen

- [Hochladen eines Objekts](#)
- [Auflisten von Objekten](#)
- [Herunterladen eines Objekts](#)
- [Kopieren, Verschieben oder Umbenennen von Objekten](#)
- [Objekte löschen](#)
- [Löschen mehrerer Objekte auf einmal](#)

### Hochladen eines Objekts

Verwenden Sie die `putObject` Methode des AmazonS3-Clients und geben Sie einen Bucket-Namen, einen Schlüsselnamen und eine Datei für den Upload an. Der Bucket muss vorhanden sein, andernfalls tritt ein Fehler auf.

### Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
```

### Code

```
System.out.format("Uploading %s to S3 bucket %s...\n", file_path, bucket_name);
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    s3.putObject(bucket_name, key_name, new File(file_path));
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Auflisten von Objekten

Um eine Liste der Objekte in einem Bucket abzurufen, verwenden Sie die `listObjects` Methode des AmazonS3-Clients und geben Sie den Namen eines Buckets an.

Die `listObjects` Methode gibt ein [ObjectListing](#) Objekt zurück, das Informationen über die Objekte im Bucket bereitstellt. Um die Objektnamen (Schlüssel) aufzulisten, verwenden Sie die `getObjectSummaries` Methode, um eine Liste von [ObjectSummaryS3-Objekten](#) abzurufen, von denen jedes ein einzelnes Objekt im Bucket darstellt. Rufen Sie dann dessen `getKey`-Methode zum Abrufen des Objektnamens auf.

### Importe

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListObjectsV2Result;
import com.amazonaws.services.s3.model.S3ObjectSummary;
```

### Code

```
System.out.format("Objects in S3 bucket %s:\n", bucket_name);
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
ListObjectsV2Result result = s3.listObjectsV2(bucket_name);
List<S3ObjectSummary> objects = result.getObjectSummaries();
for (S3ObjectSummary os : objects) {
    System.out.println("* " + os.getKey());
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Herunterladen eines Objekts

Verwenden Sie die `getObject` Methode des AmazonS3-Clients und übergeben Sie ihm den Namen eines Buckets und eines Objekts zum Herunterladen. Bei Erfolg gibt die Methode ein [S3Object](#) zurück. Der angegebene Bucket und der Objektschlüssel müssen vorhanden sein, andernfalls tritt ein Fehler auf.

Sie können den Inhalt des Objekts anfordern, indem Sie `getObjectContent` für das Objekt `S3Object` aufrufen. Dies gibt ein [S3](#) zurück `ObjectInputStream`, das sich wie ein Standard-Java-Objekt verhält. `InputStream`

Das folgende Beispiel lädt ein Objekt von S3 herunter und speichert die Inhalte in einer Datei mit dem Namen des Objektschlüssels.

### Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.S3Object;
import com.amazonaws.services.s3.model.S3ObjectInputStream;

import java.io.File;
```

### Code

```
System.out.format("Downloading %s from S3 bucket %s...\n", key_name, bucket_name);
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    S3Object o = s3.getObject(bucket_name, key_name);
    S3ObjectInputStream s3is = o.getObjectContent();
    FileOutputStream fos = new FileOutputStream(new File(key_name));
    byte[] read_buf = new byte[1024];
    int read_len = 0;
    while ((read_len = s3is.read(read_buf)) > 0) {
        fos.write(read_buf, 0, read_len);
    }
    s3is.close();
}
```

```
fos.close();
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
} catch (FileNotFoundException e) {
    System.err.println(e.getMessage());
    System.exit(1);
} catch (IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Das [vollständige Beispiel](#) finden Sie unter. [GitHub](#)

## Kopieren, Verschieben oder Umbenennen von Objekten

Sie können ein Objekt mithilfe der Methode des AmazonS3-Clients von einem Bucket in einen anderen kopieren. `copyObject` Sie nimmt den Namen des Buckets, aus dem kopiert werden soll, das zu kopierende Objekt sowie den Namen des Zielbuckets entgegen.

### Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
```

### Code

```
try {
    s3.copyObject(from_bucket, object_key, to_bucket, object_key);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
System.out.println("Done!");
```

Das [vollständige Beispiel](#) finden Sie unter. [GitHub](#)

### Note

Sie können `copyObject` mit [deleteObject](#) verwenden, um ein Objekt zu verschieben oder umzubenennen. Kopieren Sie das Objekt dazu als Erstes auf einen neuen Namen (Sie

können den gleichen Bucket als Quelle und Ziel angeben) und löschen Sie das Objekt dann von seinem bisherigen Speicherort.

## Objekte löschen

Verwenden Sie die `deleteObject` Methode des AmazonS3-Clients und übergeben Sie ihm den Namen eines Buckets und eines Objekts, das gelöscht werden soll. Der angegebene Bucket und der Objektschlüssel müssen vorhanden sein, andernfalls tritt ein Fehler auf.

### Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
```

### Code

```
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    s3.deleteObject(bucket_name, object_key);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Löschen mehrerer Objekte auf einmal

Mit der `deleteObjects` Methode des AmazonS3-Clients können Sie mehrere Objekte aus demselben Bucket löschen, indem Sie ihre Namen an die Methode [link: sdk-for-java/v1/reference/com/amazonaws/services/s3/model/DeleteObjectsRequest.html](https://docs.aws.amazon.com/sdk-for-java/v1/reference/com/amazonaws/services/s3/model/DeleteObjectsRequest.html) übergeben.

### Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
```

## Code

```
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    DeleteObjectsRequest dor = new DeleteObjectsRequest(bucket_name)
        .withKeys(object_keys);
    s3.deleteObjects(dor);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Amazon S3 Zugriffsberechtigungen für Buckets und Objekte verwalten

Sie können Zugriffskontrolllisten (ACLs) für Amazon S3 Buckets und Objekte verwenden, um Ihre Ressourcen detailliert zu steuern. Amazon S3

### Note

Bei diesen Codebeispielen wird vorausgesetzt, dass Sie die Informationen [unter Verwenden von](#) verstehen AWS SDK für Java und AWS Standardanmeldedaten anhand der Informationen unter [AWS Anmeldeinformationen einrichten und Region](#) für die Entwicklung konfiguriert haben.

## Abrufen der Zugriffskontrollliste für einen Bucket

Um die aktuelle ACL für einen Bucket abzurufen, rufen Sie die `getBucketAcl` Methode von `AmazonS3` auf und übergeben ihr den Bucket-Namen für die Abfrage. Diese Methode gibt ein [AccessControlList](#) Objekt zurück. Um jede Zugriffsberechtigung in der Liste abzurufen, rufen Sie die `getGrantsAsList`-Methode des Objekts auf. Sie erhalten dann eine Standard-Java-Liste mit [Grant](#)-Objekten.

## Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
```

```
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AccessControlList;
import com.amazonaws.services.s3.model.Grant;
```

## Code

```
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    AccessControlList acl = s3.getBucketAcl(bucket_name);
    List<Grant> grants = acl.getGrantsAsList();
    for (Grant grant : grants) {
        System.out.format("  %s: %s\n", grant.getGrantee().getIdentifier(),
            grant.getPermission().toString());
    }
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Festlegen der Zugriffskontrollliste für einen Bucket

Um einer ACL für einen Bucket Berechtigungen hinzuzufügen oder zu ändern, rufen Sie die Methode von `setBucketAcl` `AmazonS3` auf. Sie benötigt ein [AccessControlList](#) Objekt, das eine Liste von Empfängern und Zugriffsebenen für die Einrichtung enthält.

### Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AccessControlList;
import com.amazonaws.services.s3.model.EmailAddressGrantee;
```

## Code

```
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
```

```
try {
    // get the current ACL
    AccessControlList acl = s3.getBucketAcl(bucket_name);
    // set access for the grantee
    EmailAddressGrantee grantee = new EmailAddressGrantee(email);
    Permission permission = Permission.valueOf(access);
    acl.grantPermission(grantee, permission);
    s3.setBucketAcl(bucket_name, acl);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

### Note

Sie können die eindeutige ID des Empfängers direkt mithilfe der Klasse [Stipendiat](#) angeben oder die Klasse verwenden, um den [EmailAddressGrantee](#) Empfänger per E-Mail festzulegen, wie wir es hier getan haben.

Das [vollständige](#) Beispiel finden Sie unter. [GitHub](#)

## Abrufen der Zugriffskontrollliste für ein Objekt

Um die aktuelle ACL für ein Objekt abzurufen, rufen Sie die `getObjectAcl` Methode von `AmazonS3` auf und übergeben ihr den Bucket-Namen und den Objektnamen für die Abfrage. [Diese Methode gibt zum Beispiel `getBucketAcl` ein `AccessControlList` Objekt zurück, mit dem Sie jeden Grant untersuchen können.](#)

### Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AccessControlList;
import com.amazonaws.services.s3.model.Grant;
```

### Code

```
try {
```

```
AccessControlList acl = s3.getObjectAcl(bucket_name, object_key);
List<Grant> grants = acl.getGrantsAsList();
for (Grant grant : grants) {
    System.out.format("  %s: %s\n", grant.getGrantee().getIdentifizier(),
        grant.getPermission().toString());
}
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Festlegen der Zugriffskontrollliste für ein Objekt

Um einer ACL für ein Objekt Berechtigungen hinzuzufügen oder zu ändern, rufen Sie die Methode von `setObjectAcl` AmazonS3 auf. Sie benötigt ein [AccessControlList](#) Objekt, das eine Liste von Empfängern und Zugriffsebenen für die Einrichtung enthält.

### Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AccessControlList;
import com.amazonaws.services.s3.model.EmailAddressGrantee;
```

### Code

```
try {
    // get the current ACL
    AccessControlList acl = s3.getObjectAcl(bucket_name, object_key);
    // set access for the grantee
    EmailAddressGrantee grantee = new EmailAddressGrantee(email);
    Permission permission = Permission.valueOf(access);
    acl.grantPermission(grantee, permission);
    s3.setObjectAcl(bucket_name, object_key, acl);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

```
}
```

### Note

Sie können die eindeutige ID des Empfängers direkt mithilfe der Klasse [Stipendiat](#) angeben oder die Klasse verwenden, um den [EmailAddressGrantee](#) Empfänger per E-Mail festzulegen, wie wir es hier getan haben.

Das [vollständige](#) Beispiel finden Sie unter [GitHub](#)

## Weitere Informationen

- [GET Bucket acl](#) in der Amazon S3 API-Referenz
- [Fügen Sie Bucket ACL](#) in die Amazon S3 API-Referenz ein
- [GET Object acl](#) in der Amazon S3 API-Referenz
- [PUT Object acl](#) in der Amazon S3 API-Referenz

## Verwaltung des Zugriffs auf Amazon S3 Buckets mithilfe von Bucket-Richtlinien

Sie können eine Bucket-Richtlinie einrichten, abrufen oder löschen, um den Zugriff auf Ihre Amazon S3 Buckets zu verwalten.

### Festlegen einer Bucket-Richtlinie

Sie können die Bucket-Richtlinie für einen bestimmten S3-Bucket wie folgt festlegen:

- Rufen Sie den AmazonS3-Client an `setBucketPolicy` und stellen Sie ihm eine [SetBucketPolicyRequest](#)
- Durch direktes Festlegen der Richtlinie unter Verwendung der `setBucketPolicy`-Überladung, die einen Bucket-Namen und einen Richtlinientext (im JSON-Format) entgegen nimmt

### Importe

```
import com.amazonaws.AmazonServiceException;  
import com.amazonaws.auth.policy.Policy;
```

```
import com.amazonaws.auth.policy.Principal;
```

## Code

```
s3.setBucketPolicy(bucket_name, policy_text);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

## Verwenden der Policy-Klasse zum Generieren oder Überprüfen einer Richtlinie

Wenn Sie `setBucketPolicy` eine Bucket-Richtlinie übergeben, können Sie die folgenden Aufgaben ausführen:

- Direktes Übergeben der Richtlinie als Zeichenfolge mit Text im JSON-Format
- Erstellen der Richtlinie mit der [Policy](#)-Klasse

Bei Verwendung der `Policy`-Klasse müssen Sie sich keine Gedanken über die korrekte Formatierung Ihrer Text-Zeichenfolge machen. Sie können die Richtlinie als JSON-Text von der `Policy`-Klasse erhalten, indem Sie die `toJson`-Methode aufrufen.

## Importe

```
import com.amazonaws.auth.policy.Resource;
import com.amazonaws.auth.policy.Statement;
import com.amazonaws.auth.policy.actions.S3Actions;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
```

## Code

```
new Statement(Statement.Effect.Allow)
    .withPrincipals(Principal.AllUsers)
    .withActions(S3Actions.GetObject)
    .withResources(new Resource(
        "{region-arn}s3:::" + bucket_name + "/*"));
return bucket_policy.toJson();
```

Die `Policy`-Klasse bietet außerdem eine `fromJson`-Methode, mit der versucht werden kann, eine Richtlinie aus einer übergebenen JSON-Zeichenfolge zu erstellen. Die Methode überprüft die Zeichenfolge und stellt so sicher, dass sich der Text in eine gültige Richtlinienstruktur umwandeln lässt. Sie löst einen Fehler mit einer `IllegalArgumentException` aus, wenn der Richtlinientext ungültig ist.

```
Policy bucket_policy = null;
try {
    bucket_policy = Policy.fromJson(file_text.toString());
} catch (IllegalArgumentException e) {
    System.out.format("Invalid policy text in file: \"%s\"",
        policy_file);
    System.out.println(e.getMessage());
}
```

Mit dieser Technik können Sie eine Richtlinie im Voraus validieren, die Sie aus einer Datei oder anderweitig einlesen.

Das [vollständige Beispiel](#) finden Sie unter. [GitHub](#)

## Abrufen einer Bucket-Richtlinie

Um die Richtlinie für einen Amazon S3 Bucket abzurufen, rufen Sie die `getBucketPolicy` Methode des `AmazonS3`-Clients auf und übergeben Sie ihr den Namen des Buckets, aus dem die Richtlinie abgerufen werden soll.

### Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
```

### Code

```
try {
    BucketPolicy bucket_policy = s3.getBucketPolicy(bucket_name);
    policy_text = bucket_policy.getPolicyText();
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
}
```

```
    System.exit(1);  
}
```

Wenn der angegebene Bucket nicht vorhanden ist, Sie nicht darauf zugreifen können oder wenn keine Bucket-Richtlinie eingerichtet ist, wird eine `AmazonServiceException` ausgelöst.

Das [vollständige Beispiel](#) finden Sie unter. [GitHub](#)

## Löschen einer Bucket-Richtlinie

Um eine Bucket-Richtlinie zu löschen, rufen Sie den `AmazonS3`-Client auf und geben Sie ihm den Bucket-Namen. `deleteBucketPolicy`

### Importe

```
import com.amazonaws.AmazonServiceException;  
import com.amazonaws.regions.Regions;  
import com.amazonaws.services.s3.AmazonS3;
```

### Code

```
try {  
    s3.deleteBucketPolicy(bucket_name);  
} catch (AmazonServiceException e) {  
    System.err.println(e.getMessage());  
    System.exit(1);  
}
```

Diese Methode wird auch dann erfolgreich ausgeführt, wenn der Bucket noch nicht über eine Richtlinie verfügt. Wenn Sie den Namen eines Buckets angeben, der noch nicht vorhanden ist oder für den Sie keinen Zugriff haben, wird eine `AmazonServiceException` ausgelöst.

Das [vollständige Beispiel](#) finden Sie unter. [GitHub](#)

## Weitere Infos

- [Überblick über die Sprache der Access Policy](#) im Amazon Simple Storage Service Benutzerhandbuch
- [Beispiele für Bucket-Richtlinien](#) im Amazon Simple Storage Service Benutzerhandbuch

## TransferManager Für Amazon S3 Operationen verwenden

Sie können die AWS SDK für Java TransferManager Klasse verwenden, um Dateien zuverlässig aus der lokalen Umgebung zu übertragen Amazon S3 und Objekte von einem S3-Speicherort an einen anderen zu kopieren. TransferManager kann den Fortschritt einer Übertragung abrufen und Uploads und Downloads anhalten oder fortsetzen.

### Note

#### Bewährte Methode

Wir empfehlen Ihnen, die [AbortIncompleteMultipartUpload](#) Lebenszyklusregel für Ihre Amazon S3 Buckets zu aktivieren.

Diese Regel weist darauf Amazon S3 hin, dass mehrteilige Uploads abgebrochen werden, die nicht innerhalb einer bestimmten Anzahl von Tagen nach der Initiierung abgeschlossen werden. Wenn das festgelegte Zeitlimit überschritten wird, wird der Upload Amazon S3 abgebrochen und anschließend die unvollständigen Upload-Daten gelöscht.

Weitere Informationen finden Sie unter [Lebenszykluskonfiguration für einen Bucket mit Versionierung](#) im Amazon S3 Benutzerhandbuch.

### Note

Bei diesen Codebeispielen wird vorausgesetzt, dass Sie die Informationen [unter Verwenden von](#) verstehen AWS SDK für Java und AWS Standardanmeldedaten anhand der Informationen unter [AWS Anmeldeinformationen einrichten und Region für die Entwicklung](#) konfiguriert haben.

## Hochladen von Dateien und Verzeichnissen

TransferManager kann Dateien, Dateilisten und Verzeichnisse in alle Amazon S3 Buckets hochladen, die Sie [zuvor erstellt](#) haben.

### Themen

- [Hochladen einer einzelnen Datei](#)
- [Hochladen einer Dateiliste](#)
- [Upload eines Verzeichnisses](#)

## Hochladen einer einzelnen Datei

Rufen Sie `TransferManager` die `upload` Methode auf und geben Sie einen Amazon S3 Bucket-Namen, einen Schlüsselnamen (Objektnamen) und ein [Standard-Java-Dateiobjekt](#) an, das die hochzuladende Datei darstellt.

### Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.s3.transfer.MultipleFileUpload;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
import com.amazonaws.services.s3.transfer.Upload;

import java.io.File;
import java.util.ArrayList;
import java.util.Arrays;
```

### Code

```
File f = new File(file_path);
TransferManager xfer_mgr = TransferManagerBuilder.standard().build();
try {
    Upload xfer = xfer_mgr.upload(bucket_name, key_name, f);
    // loop with Transfer.isDone()
    XferMgrProgress.showTransferProgress(xfer);
    // or block with Transfer.waitForCompletion()
    XferMgrProgress.waitForCompletion(xfer);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
xfer_mgr.shutdownNow();
```

Die `upload`-Methode kehrt sofort zurück und stellt ein `Upload`-Objekt bereit, mit dem Sie den Übertragungsstatus abrufen oder auf die Fertigstellung warten können.

Informationen [dazu, wie Sie eine Übertragung vor dem Aufrufen der `shutdownNow` Methode erfolgreich abschließen `waitForCompletion` können, finden Sie unter `Warten`](#), bis eine Übertragung abgeschlossen `TransferManager` ist. Während Sie darauf warten, dass die Übertragung abgeschlossen ist, können Sie Aktualisierungen zum Status und Fortschritt abfragen oder diese als

Ereignisse empfangen. Weitere Informationen finden Sie unter [Abrufen des Übertragungsstatus und -fortschritts](#).

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Hochladen einer Dateiliste

Sie können mehrere Dateien auf einmal hochladen, indem Sie die TransferManager-Methode des `uploadFileList` aufrufen und dabei Folgendes angeben:

- Ein Amazon S3 Bucket-Name
- Schlüsselpräfix, das dem Namen der erstellten Objekte vorangestellt wird (Pfad innerhalb des Buckets, wo die Objekte abgespeichert werden sollen)
- [File](#)-Objekt, das das relative Verzeichnis darstellt, von dem aus die Dateipfade erstellt werden sollen
- [List](#)-Objekt mit einer Reihe von [File](#)-Objekten zum Hochladen

## Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.s3.transfer.MultipleFileUpload;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
import com.amazonaws.services.s3.transfer.Upload;

import java.io.File;
import java.util.ArrayList;
import java.util.Arrays;
```

## Code

```
ArrayList<File> files = new ArrayList<File>();
for (String path : file_paths) {
    files.add(new File(path));
}

TransferManager xfer_mgr = TransferManagerBuilder.standard().build();
try {
    MultipleFileUpload xfer = xfer_mgr.uploadFileList(bucket_name,
```

```
        key_prefix, new File("."), files);
    // loop with Transfer.isDone()
    XferMgrProgress.showTransferProgress(xfer);
    // or block with Transfer.waitForCompletion()
    XferMgrProgress.waitForCompletion(xfer);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
xfer_mgr.shutdownNow();
```

Informationen dazu, wie Sie [die shutdownNow Methode verwenden, um eine Übertragung vor dem Aufrufen TransferManager erfolgreich abzuschließen](#)[waitForCompletion](#), finden Sie unter [Warten](#) auf den Abschluss einer Übertragung. Während Sie darauf warten, dass die Übertragung abgeschlossen ist, können Sie Aktualisierungen zum Status und Fortschritt abfragen oder diese als Ereignisse empfangen. Weitere Informationen finden Sie unter [Abrufen des Übertragungsstatus und -fortschritts](#).

Das von zurückgegebene [MultipleFileUpload](#) Objekt `uploadFileList` kann verwendet werden, um den Status oder den Fortschritt der Übertragung abzufragen. Weitere [Informationen finden Sie unter Den aktuellen Status einer Übertragung abfragen und ProgressListener Übertragungsstatus abrufen mit a](#).

Sie können auch die `MultipleFileUpload`-Methode der `getSubTransfers`-Klasse verwenden, um die einzelnen Upload-Objekte für jede zu übertragende Datei zu erhalten. Weitere Informationen finden Sie unter [Abruf des Fortschritts von untergeordneten Übertragungen](#).

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Upload eines Verzeichnisses

Sie können die `uploadDirectory` Methode verwenden `TransferManager`, um ein ganzes Verzeichnis von Dateien hochzuladen, mit der Option, Dateien in Unterverzeichnisse rekursiv zu kopieren. Sie geben einen Amazon S3 Bucket-Namen, ein S3-Schlüsselpräfix, ein [File-Objekt](#), das das zu kopierende lokale Verzeichnis darstellt, und einen `boolean` Wert an, der angibt, ob Sie Unterverzeichnisse rekursiv kopieren möchten (`true` oder `false`).

## Importe

```
import com.amazonaws.AmazonServiceException;
```

```
import com.amazonaws.services.s3.transfer.MultipleFileUpload;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
import com.amazonaws.services.s3.transfer.Upload;

import java.io.File;
import java.util.ArrayList;
import java.util.Arrays;
```

## Code

```
TransferManager xfer_mgr = TransferManagerBuilder.standard().build();
try {
    MultipleFileUpload xfer = xfer_mgr.uploadDirectory(bucket_name,
        key_prefix, new File(dir_path), recursive);
    // loop with Transfer.isDone()
    XferMgrProgress.showTransferProgress(xfer);
    // or block with Transfer.waitForCompletion()
    XferMgrProgress.waitForCompletion(xfer);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
xfer_mgr.shutdownNow();
```

Informationen dazu, wie Sie [eine Übertragung vor dem Aufrufen TransferManager der Methode erfolgreich abschließen](#) `waitForCompletion` können, finden Sie unter [Warten](#) auf den Abschluss einer Übertragung. `shutdownNow` Während Sie darauf warten, dass die Übertragung abgeschlossen ist, können Sie Aktualisierungen zum Status und Fortschritt abfragen oder diese als Ereignisse empfangen. Weitere Informationen finden Sie unter [Abrufen des Übertragungsstatus und -fortschritts](#).

Das von zurückgegebene [MultipleFileUpload](#) Objekt `uploadFileList` kann verwendet werden, um den Status oder den Fortschritt der Übertragung abzufragen. Weitere [Informationen finden Sie unter Den aktuellen Status einer Übertragung abfragen und ProgressListener Übertragungsstatus abrufen mit a](#).

Sie können auch die `MultipleFileUpload`-Methode der `getSubTransfers`-Klasse verwenden, um die einzelnen `Upload`-Objekte für jede zu übertragende Datei zu erhalten. Weitere Informationen finden Sie unter [Abruf des Fortschritts von untergeordneten Übertragungen](#).

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#).

## Herunterladen von Dateien oder Verzeichnissen

Verwenden Sie die `TransferManager` Klasse, um entweder eine einzelne Datei (Amazon S3 Objekt) oder ein Verzeichnis (ein Amazon S3 Bucket-Name gefolgt von einem Objektpräfix) herunterzuladen Amazon S3.

### Themen

- [Herunterladen einer einzelnen Datei](#)
- [Herunterladen eines Verzeichnisses](#)

### Herunterladen einer einzelnen Datei

Verwenden Sie die `download` Methode `TransferManager`'s und geben Sie den Amazon S3 Bucket-Namen an, der das Objekt enthält, das Sie herunterladen möchten, den Schlüssel- (Objekt-) Namen und ein [File-Objekt](#), das die Datei darstellt, die auf Ihrem lokalen System erstellt werden soll.

### Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.s3.transfer.Download;
import com.amazonaws.services.s3.transfer.MultipleFileDownload;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;

import java.io.File;
```

### Code

```
File f = new File(file_path);
TransferManager xfer_mgr = TransferManagerBuilder.standard().build();
try {
    Download xfer = xfer_mgr.download(bucket_name, key_name, f);
    // loop with Transfer.isDone()
    XferMgrProgress.showTransferProgress(xfer);
    // or block with Transfer.waitForCompletion()
    XferMgrProgress.waitForCompletion(xfer);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

```
xfer_mgr.shutdownNow();
```

Informationen [dazu, wie Sie eine Übertragung vor dem Aufrufen TransferManager der shutdownNow Methode erfolgreich abschließen waitForCompletion können, finden Sie unter Warten](#) auf den Abschluss einer Übertragung. Während Sie darauf warten, dass die Übertragung abgeschlossen ist, können Sie Aktualisierungen zum Status und Fortschritt abfragen oder diese als Ereignisse empfangen. Weitere Informationen finden Sie unter [Abrufen des Übertragungsstatus und -fortschritts](#).

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Herunterladen eines Verzeichnisses

Verwenden Sie die TransferManager downloadDirectory Methode, um eine Reihe von Dateien herunterzuladen, die ein gemeinsames key prefix haben (analog zu einem Verzeichnis in einem Dateisystem). Amazon S3 Die Methode verwendet den Amazon S3 Bucket-Namen, der die Objekte enthält, die Sie herunterladen möchten, das Objektpräfix, das von allen Objekten gemeinsam genutzt wird, und ein [File-Objekt](#), das das Verzeichnis darstellt, in das die Dateien auf Ihrem lokalen System heruntergeladen werden sollen. Wenn das angegebene Verzeichnis noch nicht vorhanden ist, wird es erstellt.

## Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.s3.transfer.Download;
import com.amazonaws.services.s3.transfer.MultipleFileDownload;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;

import java.io.File;
```

## Code

```
TransferManager xfer_mgr = TransferManagerBuilder.standard().build();

try {
    MultipleFileDownload xfer = xfer_mgr.downloadDirectory(
        bucket_name, key_prefix, new File(dir_path));
    // loop with Transfer.isDone()
    XferMgrProgress.showTransferProgress(xfer);
    // or block with Transfer.waitForCompletion()
    XferMgrProgress.waitForCompletion(xfer);
} catch (AmazonServiceException e) {
```

```
System.err.println(e.getMessage());
System.exit(1);
}
xfer_mgr.shutdownNow();
```

Informationen [dazu, wie Sie eine Übertragung vor dem Aufrufen TransferManager der shutdownNow Methode erfolgreich abschließen waitForCompletion können, finden Sie unter Warten](#) auf den Abschluss einer Übertragung. Während Sie darauf warten, dass die Übertragung abgeschlossen ist, können Sie Aktualisierungen zum Status und Fortschritt abfragen oder diese als Ereignisse empfangen. Weitere Informationen finden Sie unter [Abrufen des Übertragungsstatus und -fortschritts](#).

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Kopieren von Objekten

Rufen Sie die copy-Methode von TransferManager auf, um ein Objekt von einem S3-Bucket in einen anderen zu kopieren.

### Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.s3.transfer.Copy;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
```

### Code

```
System.out.println("Copying s3 object: " + from_key);
System.out.println("    from bucket: " + from_bucket);
System.out.println("    to s3 object: " + to_key);
System.out.println("    in bucket: " + to_bucket);

TransferManager xfer_mgr = TransferManagerBuilder.standard().build();
try {
    Copy xfer = xfer_mgr.copy(from_bucket, from_key, to_bucket, to_key);
    // loop with Transfer.isDone()
    XferMgrProgress.showTransferProgress(xfer);
    // or block with Transfer.waitForCompletion()
    XferMgrProgress.waitForCompletion(xfer);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

```

}
xfer_mgr.shutdownNow();

```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Warten auf die Fertigstellung einer Übertragung

Wenn Ihre Anwendung (oder Ihr Thread) blockieren kann, bis die Übertragung abgeschlossen ist, können Sie die `waitForCompletion` Methode der [Transfer-Schnittstelle](#) verwenden, um zu blockieren, bis die Übertragung abgeschlossen ist oder eine Ausnahme auftritt.

```

try {
    xfer.waitForCompletion();
} catch (AmazonServiceException e) {
    System.err.println("Amazon service error: " + e.getMessage());
    System.exit(1);
} catch (AmazonClientException e) {
    System.err.println("Amazon client error: " + e.getMessage());
    System.exit(1);
} catch (InterruptedException e) {
    System.err.println("Transfer interrupted: " + e.getMessage());
    System.exit(1);
}

```

Sie erhalten den Fortschritt der Übertragungen, wenn Sie vor dem Aufrufen Ereignisse abfragen `waitForCompletion`, einen Abfragemechanismus in einem separaten Thread implementieren oder Fortschrittsaktualisierungen asynchron mit einem empfangen. [ProgressListener](#)

Das [vollständige Beispiel](#) finden Sie unter. GitHub

## Abrufen des Übertragungsstatus und -fortschritt

Jede der von den `copy` Methoden `TransferManager` `upload*``download*`, und zurückgegebenen Klassen gibt eine Instanz einer der folgenden Klassen zurück, je nachdem, ob es sich um eine Operation mit einer oder mehreren Dateien handelt.

Klasse	Zurückgegeben von
<a href="#">Copy</a>	<code>copy</code>
<a href="#">Download</a>	<code>download</code>

Klasse	Zurückgegeben von
<a href="#">MultipleFileDownload</a>	downloadDirectory
<a href="#">Hochladen</a>	upload
<a href="#">MultipleFileUpload</a>	uploadFileList , uploadDirectory

Alle diese Klassen implementieren die [Transfer](#)-Schnittstelle. Transfer liefert nützliche Methoden, um den Fortschritt einer Übertragung abzurufen, die Übertragung zu pausieren oder fortzusetzen sowie den aktuellen oder abschließenden Status der Übertragung abzurufen.

## Themen

- [Abfragen des aktuellen Fortschritts einer Übertragung](#)
- [Holen Sie sich den Übertragungsfortschritt mit einem ProgressListener](#)
- [Abruf des Fortschritts von untergeordneten Übertragungen](#)

## Abfragen des aktuellen Fortschritts einer Übertragung

Diese Schleife gibt den Fortschritt einer Übertragung aus, untersucht den aktuellen Fortschritt während der Ausführung und gibt beim Abschluss den abschließenden Status aus.

## Importe

```
import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;
import com.amazonaws.event.ProgressEvent;
import com.amazonaws.event.ProgressListener;
import com.amazonaws.services.s3.transfer.*;
import com.amazonaws.services.s3.transfer.Transfer.TransferState;

import java.io.File;
import java.util.ArrayList;
import java.util.Collection;
```

## Code

```
// print the transfer's human-readable description
System.out.println(xfer.getDescription());
```

```
// print an empty progress bar...
printProgressBar(0.0);
// update the progress bar while the xfer is ongoing.
do {
    try {
        Thread.sleep(100);
    } catch (InterruptedException e) {
        return;
    }
    // Note: so_far and total aren't used, they're just for
    // documentation purposes.
    TransferProgress progress = xfer.getProgress();
    long so_far = progress.getBytesTransferred();
    long total = progress.getTotalBytesToTransfer();
    double pct = progress.getPercentTransferred();
    eraseProgressBar();
    printProgressBar(pct);
} while (xfer.isDone() == false);
// print the final state of the transfer.
TransferState xfer_state = xfer.getState();
System.out.println(": " + xfer_state);
```

[Das vollständige Beispiel finden Sie unter. GitHub](#)

Holen Sie sich den Übertragungsfortschritt mit einem `ProgressListener`

Mithilfe der `addProgressListener` Methode der Transfer-Schnittstelle können Sie jeder [Übertragung](#) eine [ProgressListener](#) hinzufügen.

A [ProgressListener](#) benötigt nur eine Methode `progressChanged`, die ein [ProgressEvent](#) Objekt akzeptiert. Mit diesem Objekt können Sie die Gesamtzahl der Bytes der Operation ermitteln, indem Sie die `getBytes`-Methode aufrufen. Die Gesamtzahl der übertragenen Bytes erfahren Sie mit Aufruf von `getBytesTransferred`.

Importe

```
import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;
import com.amazonaws.event.ProgressEvent;
import com.amazonaws.event.ProgressListener;
import com.amazonaws.services.s3.transfer.*;
import com.amazonaws.services.s3.transfer.Transfer.TransferState;
```

```
import java.io.File;
import java.util.ArrayList;
import java.util.Collection;
```

## Code

```
File f = new File(file_path);
TransferManager xfer_mgr = TransferManagerBuilder.standard().build();
try {
    Upload u = xfer_mgr.upload(bucket_name, key_name, f);
    // print an empty progress bar...
    printProgressBar(0.0);
    u.addProgressListener(new ProgressListener() {
        public void progressChanged(ProgressEvent e) {
            double pct = e.getBytesTransferred() * 100.0 / e.getBytes();
            eraseProgressBar();
            printProgressBar(pct);
        }
    });
    // block with Transfer.waitForCompletion()
    XferMgrProgress.waitForCompletion(u);
    // print the final state of the transfer.
    TransferState xfer_state = u.getState();
    System.out.println(": " + xfer_state);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
xfer_mgr.shutdownNow();
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Abruf des Fortschritts von untergeordneten Übertragungen

Die [MultipleFileUpload](#)-Klasse kann Informationen über ihre Unterübertragungen zurückgeben, indem sie ihre `getSubTransfers`-Methode aufruft. Sie gibt eine unveränderbare [Sammlung](#) von [Upload-Objekten](#) zurück, die den individuellen Übertragungsstatus und den Fortschritt jeder Unterübertragung angeben.

## Importe

```
import com.amazonaws.AmazonClientException;
```

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.event.ProgressEvent;
import com.amazonaws.event.ProgressListener;
import com.amazonaws.services.s3.transfer.*;
import com.amazonaws.services.s3.transfer.Transfer.TransferState;

import java.io.File;
import java.util.ArrayList;
import java.util.Collection;
```

## Code

```
Collection<? extends Upload> sub_xfers = new ArrayList<Upload>();
sub_xfers = multi_upload.getSubTransfers();

do {
    System.out.println("\nSubtransfer progress:\n");
    for (Upload u : sub_xfers) {
        System.out.println("  " + u.getDescription());
        if (u.isDone()) {
            TransferState xfer_state = u.getState();
            System.out.println("  " + xfer_state);
        } else {
            TransferProgress progress = u.getProgress();
            double pct = progress.getPercentTransferred();
            printProgressBar(pct);
            System.out.println();
        }
    }
}

// wait a bit before the next update.
try {
    Thread.sleep(200);
} catch (InterruptedException e) {
    return;
}
} while (multi_upload.isDone() == false);
// print the final state of the transfer.
TransferState xfer_state = multi_upload.getState();
System.out.println("\nMultipleFileUpload " + xfer_state);
```

Das [vollständige](#) Beispiel finden Sie unter. [GitHub](#)

## Weitere Infos

- [Objektschlüssel](#) im Amazon Simple Storage Service Benutzerhandbuch

## Einen Amazon S3 Bucket als Website konfigurieren

Sie können einen Amazon S3 Bucket so konfigurieren, dass er sich wie eine Website verhält. Hierzu müssen Sie die Website-Konfiguration festlegen.

### Note

Bei diesen Codebeispielen wird vorausgesetzt, dass Sie die Informationen [unter Verwenden von](#) verstehen AWS SDK für Java und AWS Standardanmeldedaten anhand der Informationen unter [AWS Anmeldeinformationen einrichten und Region für die Entwicklung](#) konfiguriert haben.

## Festlegen der Website-Konfiguration eines Buckets

Um die Website-Konfiguration eines Amazon S3 Buckets festzulegen, rufen Sie die `setWebsiteConfiguration` AmazonS3-Methode mit dem Bucket-Namen auf, für den die Konfiguration festgelegt werden soll, und einem [BucketWebsiteConfiguration](#)-Objekt, das die Website-Konfiguration des Buckets enthält.

Das Festlegen eines Index-Dokuments ist erforderlich. Alle anderen Parameter sind optional.

## Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketWebsiteConfiguration;
```

## Code

```
String bucket_name, String index_doc, String error_doc) {
    BucketWebsiteConfiguration website_config = null;

    if (index_doc == null) {
```

```
        website_config = new BucketWebsiteConfiguration();
    } else if (error_doc == null) {
        website_config = new BucketWebsiteConfiguration(index_doc);
    } else {
        website_config = new BucketWebsiteConfiguration(index_doc, error_doc);
    }

    final AmazonS3 s3 =
        AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
    try {
        s3.setBucketWebsiteConfiguration(bucket_name, website_config);
    } catch (AmazonServiceException e) {
        System.out.format(
            "Failed to set website configuration for bucket '%s'!\n",
            bucket_name);
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

### Note

Beim Festlegen einer Website-Konfiguration werden die Zugriffsberechtigungen für den Bucket nicht geändert. Um die enthaltenen Dateien im Internet sichtbar zu machen, müssen Sie zusätzlich eine Bucket-Richtlinie festlegen, durch die der öffentliche Lesezugriff für die Dateien in dem Bucket ermöglicht wird. Weitere Informationen finden Sie unter [Zugriff auf Amazon S3 Buckets mithilfe von Bucket-Richtlinien verwalten](#).

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Abruf der Website-Konfiguration eines Buckets

Um die Website-Konfiguration eines Amazon S3 Buckets abzurufen, rufen Sie die `getWebsiteConfiguration` AmazonS3-Methode mit dem Namen des Buckets auf, für den die Konfiguration abgerufen werden soll.

Die Konfiguration wird als Objekt zurückgegeben. [BucketWebsiteConfiguration](#) Wenn keine Website-Konfiguration für den Bucket vorhanden ist, wird `null` zurückgegeben.

### Importe

```
import com.amazonaws.AmazonServiceException;
```

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketWebsiteConfiguration;
```

## Code

```
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    BucketWebsiteConfiguration config =
        s3.getBucketWebsiteConfiguration(bucket_name);
    if (config == null) {
        System.out.println("No website configuration found!");
    } else {
        System.out.format("Index document: %s\n",
            config.getIndexDocumentSuffix());
        System.out.format("Error document: %s\n",
            config.getErrorDocument());
    }
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.out.println("Failed to get website configuration!");
    System.exit(1);
}
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

## Löschen der Website-Konfiguration eines Buckets

Um die Website-Konfiguration eines Amazon S3 Buckets zu löschen, rufen Sie die `deleteWebsiteConfiguration` Methode von `AmazonS3` mit dem Namen des Buckets auf, aus dem die Konfiguration gelöscht werden soll.

## Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
```

## Code

```
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    s3.deleteBucketWebsiteConfiguration(bucket_name);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.out.println("Failed to delete website configuration!");
    System.exit(1);
}
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Weitere Informationen

- [PUT Bucket-Website](#) in der Amazon S3 API-Referenz
- [GET Bucket-Website](#) in der Amazon S3 API-Referenz
- [DELETE Bucket-Website](#) in der Amazon S3 API-Referenz

## Amazon S3 Clientseitige Verschlüsselung verwenden

Das Verschlüsseln von Daten mit dem Amazon S3 Verschlüsselungsclient ist eine Möglichkeit, eine zusätzliche Schutzebene für vertrauliche Informationen bereitzustellen, in denen Sie gespeichert sind. Amazon S3 Die Beispiele in diesem Abschnitt zeigen, wie Sie den Amazon S3 Verschlüsselungsclient für Ihre Anwendung erstellen und konfigurieren.

Wenn Sie mit Kryptografie noch nicht vertraut sind, finden Sie in den [Grundlagen der Kryptografie](#) im AWS KMS-Entwicklerhandbuch einen grundlegenden Überblick über Begriffe und Algorithmen der Kryptografie. Informationen zur allgemeinen Kryptografieunterstützung finden Sie AWS SDKs unter [AWS SDK-Unterstützung für Amazon S3 clientseitige Verschlüsselung](#) in der Amazon Web Services Allgemeinen Referenz.

### Note

Bei diesen Codebeispielen wird vorausgesetzt, dass Sie die Informationen [unter Verwenden von](#) verstehen AWS SDK für Java und AWS Standardanmeldedaten anhand der Informationen unter [AWS Anmeldeinformationen einrichten und Region](#) für die Entwicklung konfiguriert haben.

Wenn Sie Version 1.11.836 oder eine frühere Version von verwenden AWS SDK für Java, finden Sie unter [Amazon S3 Encryption Client Migration Informationen zur Migration](#) Ihrer Anwendungen auf spätere Versionen. Wenn Sie nicht migrieren können, finden Sie [dieses](#) vollständige Beispiel unter GitHub

Wenn Sie Version 1.11.837 oder höher von verwenden AWS SDK für Java, sollten Sie sich andernfalls die unten aufgeführten Beispielthemen zur Verwendung Amazon S3 der clientseitigen Verschlüsselung ansehen.

## Themen

- [Amazon S3 clientseitige Verschlüsselung mit Client-Hauptschlüsseln](#)
- [Amazon S3 clientseitige Verschlüsselung mit AWS KMS-verwalteten Schlüsseln](#)

## Amazon S3 clientseitige Verschlüsselung mit Client-Hauptschlüsseln

In den folgenden Beispielen wird die [AmazonS3 EncryptionClient V2Builder-Klasse](#) verwendet, um einen Amazon S3 Client mit aktivierter clientseitiger Verschlüsselung zu erstellen. Nach der Aktivierung werden alle Objekte, auf die Sie mit diesem Client hochladen, Amazon S3 verschlüsselt. Alle Objekte, die Sie Amazon S3 mit diesem Client erhalten, werden automatisch entschlüsselt.

### Note

Die folgenden Beispiele zeigen die Verwendung der Amazon S3 clientseitigen Verschlüsselung mit kundenverwalteten Client-Hauptschlüsseln. Informationen zur Verwendung der Verschlüsselung mit von AWS KMS verwalteten Schlüsseln finden Sie unter [Amazon S3 Clientseitige Verschlüsselung](#) mit von KMS verwalteten Schlüsseln. AWS

Bei der Aktivierung der clientseitigen Verschlüsselung können Sie zwischen zwei Verschlüsselungsmodi wählen: strikt authentifiziert Amazon S3 oder authentifiziert. In den folgenden Abschnitten sehen Sie, wie die unterschiedlichen Modi aktiviert werden. Informationen zu den Algorithmen, die in den einzelnen Modi verwendet werden, finden Sie in der Definition. [CryptoMode](#)

## Erforderliche Importe

Importieren Sie für diese Beispiele die folgenden Klassen.

## Importe

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3EncryptionClientV2Builder;
import com.amazonaws.services.s3.AmazonS3EncryptionV2;
import com.amazonaws.services.s3.model.CryptoConfigurationV2;
import com.amazonaws.services.s3.model.CryptoMode;
import com.amazonaws.services.s3.model.EncryptionMaterials;
import com.amazonaws.services.s3.model.StaticEncryptionMaterialsProvider;
```

## Strikte authentifizierte Verschlüsselung

Strikte authentifizierte Verschlüsselung ist der Standardmodus, wenn kein Modus angegeben `CryptoMode` ist.

Um diesen Modus explizit zu aktivieren, geben Sie den `StrictAuthenticatedEncryption` Wert in der `withCryptoConfiguration` Methode an.

### Note

Bei Verwendung der clientseitigen authentifizierten Verschlüsselung müssen Sie die neueste [Bouncy Castle jar](#)-Datei im Klassenpfad Ihrer Anwendung einschließen.

## Code

```
AmazonS3EncryptionV2 s3Encryption = AmazonS3EncryptionClientV2Builder.standard()
    .withRegion(Regions.US_WEST_2)
    .withCryptoConfiguration(new
    CryptoConfigurationV2().withCryptoMode((CryptoMode.StrictAuthenticatedEncryption)))
    .withEncryptionMaterialsProvider(new StaticEncryptionMaterialsProvider(new
    EncryptionMaterials(secretKey)))
    .build();

s3Encryption.putObject(bucket_name, ENCRYPTED_KEY2, "This is the 2nd content to
encrypt");
```

## Authentifizierter Verschlüsselungsmodus

Beim Modus `AuthenticatedEncryption` wird während der Verschlüsselung ein verbesserter Schlüsselverpackungsalgorithmus angewendet. Bei einer Entschlüsselung in diesem Modus verifiziert

der Algorithmus die Integrität des entschlüsselten Objekts und löst eine Ausnahme aus, wenn das Objekt nicht verifiziert werden kann. Weitere Informationen zur Funktionsweise der authentifizierten Verschlüsselung finden Sie im Blogbeitrag [Amazon S3 Client-Side Authenticated Encryption](#).

### Note

Bei Verwendung der clientseitigen authentifizierten Verschlüsselung müssen Sie die neueste [Bouncy Castle jar](#)-Datei im Klassenpfad Ihrer Anwendung einschließen.

Zur Aktivierung des Modus geben Sie den `AuthenticatedEncryption`-Wert in der `withCryptoConfiguration`-Methode an.

### Code

```
AmazonS3EncryptionV2 s3EncryptionClientV2 =
    AmazonS3EncryptionClientV2Builder.standard()
        .withRegion(Regions.DEFAULT_REGION)
        .withClientConfiguration(new ClientConfiguration())
        .withCryptoConfiguration(new
    CryptoConfigurationV2().withCryptoMode(CryptoMode.AuthenticatedEncryption))
        .withEncryptionMaterialsProvider(new StaticEncryptionMaterialsProvider(new
    EncryptionMaterials(secretKey)))
        .build();

s3EncryptionClientV2.putObject(bucket_name, ENCRYPTED_KEY1, "This is the 1st content to
encrypt");
```

## Amazon S3 clientseitige Verschlüsselung mit AWS KMS-verwalteten Schlüsseln

In den folgenden Beispielen wird die [AmazonS3 EncryptionClient V2Builder-Klasse](#) verwendet, um einen Amazon S3 Client mit aktivierter clientseitiger Verschlüsselung zu erstellen. Nach der Konfiguration werden alle Objekte, auf die Sie mit diesem Client hochladen, Amazon S3 verschlüsselt. Alle Objekte, die Sie Amazon S3 über diesen Client erhalten, werden automatisch entschlüsselt.

### Note

Die folgenden Beispiele zeigen, wie die Amazon S3 clientseitige Verschlüsselung mit verwalteten AWS KMS-Schlüsseln verwendet wird. Informationen zur Verwendung der

Verschlüsselung mit Ihren eigenen Schlüsseln finden Sie unter [Amazon S3 Clientseitige Verschlüsselung mit Client-Hauptschlüsseln](#).

Bei der Aktivierung der clientseitigen Verschlüsselung können Sie zwischen zwei Verschlüsselungsmodi wählen: strikt authentifiziert Amazon S3 oder authentifiziert. In den folgenden Abschnitten sehen Sie, wie die unterschiedlichen Modi aktiviert werden. Informationen zu den Algorithmen, die in den einzelnen Modi verwendet werden, finden Sie in der Definition. [CryptoMode](#)

### Erforderliche Importe

Importieren Sie für diese Beispiele die folgenden Klassen.

### Importe

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.kms.AWSKMS;
import com.amazonaws.services.kms.AWSKMSClientBuilder;
import com.amazonaws.services.kms.model.GenerateDataKeyRequest;
import com.amazonaws.services.kms.model.GenerateDataKeyResult;
import com.amazonaws.services.s3.AmazonS3EncryptionClientV2Builder;
import com.amazonaws.services.s3.AmazonS3EncryptionV2;
import com.amazonaws.services.s3.model.CryptoConfigurationV2;
import com.amazonaws.services.s3.model.CryptoMode;
import com.amazonaws.services.s3.model.EncryptionMaterials;
import com.amazonaws.services.s3.model.KMSEncryptionMaterialsProvider;
```

### Strikte authentifizierte Verschlüsselung

Strikte authentifizierte Verschlüsselung ist der Standardmodus, wenn kein Modus angegeben `CryptoMode` ist.

Um diesen Modus explizit zu aktivieren, geben Sie den `StrictAuthenticatedEncryption` Wert in der `withCryptoConfiguration` Methode an.

#### Note

Bei Verwendung der clientseitigen authentifizierten Verschlüsselung müssen Sie die neueste [Bouncy Castle jar](#)-Datei im Klassenpfad Ihrer Anwendung einschließen.

## Code

```
AmazonS3EncryptionV2 s3Encryption = AmazonS3EncryptionClientV2Builder.standard()
    .withRegion(Regions.US_WEST_2)
    .withCryptoConfiguration(new
    CryptoConfigurationV2().withCryptoMode((CryptoMode.StrictAuthenticatedEncryption)))
    .withEncryptionMaterialsProvider(new KMSEncryptionMaterialsProvider(keyId))
    .build();

s3Encryption.putObject(bucket_name, ENCRYPTED_KEY3, "This is the 3rd content to encrypt
with a key created in the {console}");
System.out.println(s3Encryption.getObjectAsString(bucket_name, ENCRYPTED_KEY3));
```

Rufen Sie die `putObject` Methode auf dem Amazon S3 Verschlüsselungsclient auf, um Objekte hochzuladen.

## Code

```
s3Encryption.putObject(bucket_name, ENCRYPTED_KEY3, "This is the 3rd content to encrypt
with a key created in the {console}");
```

Sie können das Objekt mit demselben Client abrufen. Bei diesem Beispiel wird die `getObjectAsString`-Methode zum Abrufen der gespeicherten Zeichenfolge eingesetzt.

## Code

```
System.out.println(s3Encryption.getObjectAsString(bucket_name, ENCRYPTED_KEY3));
```

## Authentifizierter Verschlüsselungsmodus

Beim Modus `AuthenticatedEncryption` wird während der Verschlüsselung ein verbesserter Schlüsselverpackungsalgorithmus angewendet. Bei einer Entschlüsselung in diesem Modus verifiziert der Algorithmus die Integrität des entschlüsselten Objekts und löst eine Ausnahme aus, wenn das Objekt nicht verifiziert werden kann. Weitere Informationen zur Funktionsweise der authentifizierten Verschlüsselung finden Sie im Blogbeitrag [Amazon S3 Client-Side Authenticated Encryption](#).

### Note

Bei Verwendung der clientseitigen authentifizierten Verschlüsselung müssen Sie die neueste [Bouncy Castle jar](#)-Datei im Klassenpfad Ihrer Anwendung einschließen.

Zur Aktivierung des Modus geben Sie den `AuthenticatedEncryption`-Wert in der `withCryptoConfiguration`-Methode an.

### Code

```
AmazonS3EncryptionV2 s3Encryption = AmazonS3EncryptionClientV2Builder.standard()
    .withRegion(Regions.US_WEST_2)
    .withCryptoConfiguration(new
    CryptoConfigurationV2().withCryptoMode((CryptoMode.AuthenticatedEncryption)))
    .withEncryptionMaterialsProvider(new KMSEncryptionMaterialsProvider(keyId))
    .build();
```

### AWS KMS Den Client konfigurieren

Der Amazon S3 Verschlüsselungsclient erstellt standardmäßig einen AWS KMS Client, sofern nicht explizit einer angegeben ist.

Um die Region für diesen automatisch erstellten AWS KMS Client festzulegen, legen Sie den fest. `awsKmsRegion`

### Code

```
Region kmsRegion = Region.getRegion(Regions.AP_NORTHEAST_1);

AmazonS3EncryptionV2 s3Encryption = AmazonS3EncryptionClientV2Builder.standard()
    .withRegion(Regions.US_WEST_2)
    .withCryptoConfiguration(new
    CryptoConfigurationV2().withAwsKmsRegion(kmsRegion))
    .withEncryptionMaterialsProvider(new KMSEncryptionMaterialsProvider(keyId))
    .build();
```

Alternativ können Sie Ihren eigenen AWS KMS Client verwenden, um den Verschlüsselungsclient zu initialisieren.

### Code

```
AWSKMS kmsClient = AWSKMSClientBuilder.standard()
    .withRegion(Regions.US_WEST_2);
    .build();

AmazonS3EncryptionV2 s3Encryption = AmazonS3EncryptionClientV2Builder.standard()
```

```
.withRegion(Regions.US_WEST_2)
.withKmsClient(kmsClient)
.withCryptoConfiguration(new
CryptoConfigurationV2().withCryptoMode((CryptoMode.AuthenticatedEncryption)))
.withEncryptionMaterialsProvider(new KMSEncryptionMaterialsProvider(keyId))
.build();
```

## Amazon SQS Beispiele für die Verwendung der AWS SDK für Java

Dieser Abschnitt bietet Beispiele für die Programmierung von [Amazon SQS](#) mithilfe des [AWS SDK für Java](#).

### Note

Die 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

- [Mit Amazon SQS Nachrichtenwarteschlangen arbeiten](#)
- [Amazon SQS Nachrichten senden, empfangen und löschen](#)
- [Long Polling für Amazon SQS Nachrichtenwarteschlangen aktivieren](#)
- [Sichtbarkeits-Timeout einrichten in Amazon SQS](#)
- [Verwenden von Warteschlangen für unzustellbare Briefe in Amazon SQS](#)

## Mit Amazon SQS Nachrichtenwarteschlangen arbeiten

Eine Nachrichtenwarteschlange ist der logische Container, in den Nachrichten zuverlässig gesendet werden Amazon SQS. 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 SQS Entwicklerhandbuch](#).

In diesem Thema wird beschrieben, wie Sie mithilfe von eine Amazon SQS Warteschlange erstellen, auflisten, löschen und deren URL abrufen. AWS SDK für Java

## Erstellen einer Warteschlange

Verwenden Sie die `createQueue` Methode des AmazonSQS-Clients und stellen Sie ein [CreateQueueRequest](#) Objekt bereit, das die Warteschlangenparameter beschreibt.

### Importe

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.AmazonSQSException;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
```

### Code

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();
CreateQueueRequest create_request = new CreateQueueRequest(QueueName)
    .addAttributesEntry("DelaySeconds", "60")
    .addAttributesEntry("MessageRetentionPeriod", "86400");

try {
    sqs.createQueue(create_request);
} catch (AmazonSQSException e) {
    if (!e.getErrorCode().equals("QueueAlreadyExists")) {
        throw e;
    }
}
```

Sie können die vereinfachte Form von `createQueue` verwenden, die nur einen Namen für die Warteschlange benötigt, um eine Standard-Warteschlange zu erstellen.

```
sqs.createQueue("MyQueue" + new Date().getTime());
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Auflisten von Warteschlangen

Rufen Sie die Methode des AmazonSQS-Clients auf, um die Amazon SQS Warteschlangen für Ihr Konto aufzulisten. `listQueues`

### Importe

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.ListQueuesResult;
```

## Code

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();
ListQueuesResult lq_result = sqs.listQueues();
System.out.println("Your SQS Queue URLs:");
for (String url : lq_result.getQueueUrls()) {
    System.out.println(url);
}
```

Wenn Sie die `listQueues`-Überladung ohne Parameter aufrufen, werden alle Warteschlangen zurückgegeben. Sie können die zurückgegebenen Ergebnisse filtern, indem Sie ein `ListQueuesRequest`-Objekt übergeben.

## Importe

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.ListQueuesRequest;
```

## Code

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();
String name_prefix = "Queue";
lq_result = sqs.listQueues(new ListQueuesRequest(name_prefix));
System.out.println("Queue URLs with prefix: " + name_prefix);
for (String url : lq_result.getQueueUrls()) {
    System.out.println(url);
}
```

Das [vollständige](#) Beispiel finden Sie unter. [GitHub](#)

## Abrufen der URL für eine Warteschlange

Rufen Sie die Methode des AmazonSQS-Clients auf. `getQueueUrl`

## Importe

```
import com.amazonaws.services.sqs.AmazonSQS;  
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
```

## Code

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();  
String queue_url = sqs.getQueueUrl(QueueName).getQueueUrl();
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Löschen einer Warteschlange

Geben Sie die [URL](#) der Warteschlange für die Methode des AmazonSQS-Clients `deleteQueue`.

## Importe

```
import com.amazonaws.services.sqs.AmazonSQS;  
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
```

## Code

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();  
sqs.deleteQueue(queue_url);
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Weitere Infos

- [So funktionieren Amazon SQS Warteschlangen](#) im Amazon SQS Entwicklerhandbuch
- [CreateQueue](#) in der Amazon SQS API-Referenz
- [GetQueueUrl](#) in der Amazon SQS API-Referenz
- [ListQueues](#) in der Amazon SQS API-Referenz
- [DeleteQueues](#) in der Amazon SQS API-Referenz

## Amazon SQS Nachrichten senden, empfangen und löschen

In diesem Thema wird beschrieben, wie Sie Amazon SQS Nachrichten senden, empfangen und löschen. Nachrichten werden immer mit einer [SQS-Warteschlange](#) geliefert.

## Senden einer Nachricht

Fügen Sie einer Amazon SQS Warteschlange eine einzelne Nachricht hinzu, indem Sie die Methode des AmazonSQS-Clients aufrufen. `sendMessage` Geben Sie ein [SendMessageRequest](#) Objekt an, das die [URL](#) der Warteschlange, den Nachrichtentext und den optionalen Verzögerungswert (in Sekunden) enthält.

### Importe

```
import com.amazonaws.services.sqs.AmazonSQS;  
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;  
import com.amazonaws.services.sqs.model.SendMessageRequest;
```

### Code

```
SendMessageRequest send_msg_request = new SendMessageRequest()  
    .withQueueUrl(queueUrl)  
    .withMessageBody("hello world")  
    .withDelaySeconds(5);  
sqs.sendMessage(send_msg_request);
```

Das [vollständige Beispiel](#) finden Sie unter GitHub.

### Senden mehrerer Nachrichten gleichzeitig

Sie können mehrere Nachrichten in einer einzigen Anforderung senden. Um mehrere Nachrichten zu senden, verwenden Sie die `sendMessageBatch` Methode des AmazonSQS-Clients, die eine URL [SendMessageBatchRequest](#) mit der Warteschlange und eine Liste von Nachrichten (jeweils eine [SendMessageBatchRequestEntry](#)) zum Senden verwendet. Sie können auch eine optionale Verzögerung pro Nachricht festlegen.

### Importe

```
import com.amazonaws.services.sqs.model.SendMessageBatchRequest;  
import com.amazonaws.services.sqs.model.SendMessageBatchRequestEntry;
```

### Code

```
SendMessageBatchRequest send_batch_request = new SendMessageBatchRequest()  
    .withQueueUrl(queueUrl)  
    .withEntries(  
        ...  
    );
```

```
        new SendMessageBatchRequestEntry(
            "msg_1", "Hello from message 1"),
        new SendMessageBatchRequestEntry(
            "msg_2", "Hello from message 2")
            .withDelaySeconds(10));
sqs.sendMessageBatch(send_batch_request);
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Empfangen von Nachrichten

Rufen Sie alle Nachrichten ab, die sich derzeit in der Warteschlange befinden, indem Sie die `receiveMessage` Methode des AmazonSQS-Clients aufrufen und ihr die URL der Warteschlange übergeben. Nachrichten werden als Liste von [Message](#)-Objekten zurückgegeben.

### Importe

```
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.AmazonSQSException;
import com.amazonaws.services.sqs.model.SendMessageBatchRequest;
```

### Code

```
List<Message> messages = sqs.receiveMessage(queueUrl).getMessages();
```

## Löschen von Nachrichten nach dem Empfangen

Nachdem Sie eine Nachricht empfangen und ihren Inhalt verarbeitet haben, löschen Sie die Nachricht aus der Warteschlange, indem Sie die Empfangsnummer und die Warteschlangen-URL der Nachricht an die Methode des AmazonSQS-Clients senden. `deleteMessage`

### Code

```
for (Message m : messages) {
    sqs.deleteMessage(queueUrl, m.getReceiptHandle());
}
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Weitere Infos

- [So funktionieren Amazon SQS Warteschlangen](#) im Amazon SQS Entwicklerhandbuch

- [SendMessage](#) in der Amazon SQS API-Referenz
- [SendMessageBatch](#) in der Amazon SQS API-Referenz
- [ReceiveMessage](#) in der Amazon SQS API-Referenz
- [DeleteMessage](#) in der Amazon SQS API-Referenz

## Long Polling für Amazon SQS Nachrichtenwarteschlangen aktivieren

Amazon SQS verwendet standardmäßig kurze Abfragen, bei denen nur eine Teilmenge der Server anhand einer gewichteten Zufallsverteilung abgefragt wird, um festzustellen, ob Nachrichten für die Antwort verfügbar sind.

Lange Abfragen tragen dazu bei, die Nutzungskosten zu senken, Amazon SQS indem die Anzahl der leeren Antworten reduziert wird, wenn keine Nachrichten als Antwort auf eine an eine Warteschlange gesendete `ReceiveMessage` Anfrage zur Verfügung stehen, und falsche Leerantworten vermieden werden. Amazon SQS

### Note

Sie können eine lange Abfragefrequenz zwischen 1 und 20 Sekunden festlegen.

## Aktivieren der Langabfrage beim Erstellen einer Warteschlange

Um lange Abfragen beim Erstellen einer Amazon SQS Warteschlange zu aktivieren, legen Sie das `ReceiveMessageWaitTimeSeconds` Attribut für das [CreateQueueRequest](#) Objekt fest, bevor Sie die Methode der `AmazonSQS`-Klasse aufrufen. `createQueue`

### Importe

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.AmazonSQSException;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
```

### Code

```
final AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();
```

```
// Enable long polling when creating a queue
CreateQueueRequest create_request = new CreateQueueRequest()
    .withQueueName(queue_name)
    .addAttributesEntry("ReceiveMessageWaitTimeSeconds", "20");

try {
    sqs.createQueue(create_request);
} catch (AmazonSQSException e) {
    if (!e.getErrorCode().equals("QueueAlreadyExists")) {
        throw e;
    }
}
```

[Das vollständige Beispiel finden Sie unter. GitHub](#)

## Aktivieren der Langabfrage für eine vorhandene Warteschlange

Sie können nicht nur die lange Abfrage beim Erstellen einer Warteschlange aktivieren, sondern sie auch für eine bestehende Warteschlange aktivieren, indem Sie die Methode [SetQueueAttributesRequest](#) vor `ReceiveMessageWaitTimeSeconds` dem Aufrufen der AmazonSQS-Klasse aktivieren. `setQueueAttributes`

### Importe

```
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;
```

### Code

```
SetQueueAttributesRequest set_attrs_request = new SetQueueAttributesRequest()
    .withQueueUrl(queue_url)
    .addAttributesEntry("ReceiveMessageWaitTimeSeconds", "20");
sqs.setQueueAttributes(set_attrs_request);
```

[Das vollständige Beispiel finden Sie unter. GitHub](#)

## Aktivieren von Langabfragen beim Nachrichteneingang

Sie können lange Abfragen beim Empfang einer Nachricht aktivieren, indem Sie die Wartezeit in Sekunden für die Methode festlegen [ReceiveMessageRequest](#), die Sie an die Methode der AmazonSQS-Klasse übergeben. `receiveMessage`

**Note**

Sie sollten sicherstellen, dass das Anfrage-Timeout des AWS Kunden größer ist als die maximale lange Abfragezeit (20 s), damit es bei Ihren `receiveMessage` Anfragen nicht zu einem Timeout kommt, während Sie auf das nächste Umfrageereignis warten!

**Importe**

```
import com.amazonaws.services.sqs.model.ReceiveMessageRequest;
```

**Code**

```
ReceiveMessageRequest receive_request = new ReceiveMessageRequest()  
    .withQueueUrl(queue_url)  
    .withWaitTimeSeconds(20);  
sqs.receiveMessage(receive_request);
```

Das [vollständige Beispiel](#) finden Sie unter. GitHub

**Weitere Infos**

- [Amazon SQS Long Polling](#) im Amazon SQS Entwicklerhandbuch
- [CreateQueue](#) in der Amazon SQS API-Referenz
- [ReceiveMessage](#) in der Amazon SQS API-Referenz
- [SetQueueAttributes](#) in der Amazon SQS API-Referenz

**Sichtbarkeits-Timeout einrichten in Amazon SQS**

Wenn eine Nachricht empfangen wird Amazon SQS, verbleibt sie in der Warteschlange, bis sie gelöscht wird, um den Empfang sicherzustellen. Eine empfangene, aber nicht gelöschte Nachricht erscheint erst nach Ablauf einer bestimmten Zeitbeschränkung für die Sichtbarkeit in nachfolgenden Anforderungen. Dadurch wird gewährleistet, dass die Nachricht nicht mehrmals empfangen wird, bevor sie verarbeitet und gelöscht werden kann.

**Note**

Bei Nutzung von [Standard-Warteschlangen](#) kann durch die Zeitbeschränkung für die Sichtbarkeit nicht garantiert werden, dass eine Nachricht mehrmals empfangen wird. Wenn Sie eine Standard-Warteschlange verwenden, achten Sie darauf, dass Ihr Code mit dem Fall umgehen kann, dass dieselbe Nachricht mehrmals eingeht.

## Einrichten der Zeitbeschränkung für die Sichtbarkeit einer einzelnen Nachricht

Wenn Sie eine Nachricht erhalten haben, können Sie ihr Sichtbarkeits-Timeout ändern, indem Sie ihre Empfangsnummer in einer Methode übergeben [ChangeMessageVisibilityRequest](#), die Sie an die Methode der AmazonSQS-Klasse übergeben. `changeMessageVisibility`

### Importe

```
import com.amazonaws.services.sqs.AmazonSQS;  
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
```

### Code

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();  
  
// Get the receipt handle for the first message in the queue.  
String receipt = sqs.receiveMessage(queue_url)  
    .getMessages()  
    .get(0)  
    .getReceiptHandle();  
  
sqs.changeMessageVisibility(queue_url, receipt, timeout);
```

[Das vollständige Beispiel finden Sie unter.](#) [GitHub](#)

## Einrichten der Zeitbeschränkung für die Sichtbarkeit mehrerer Nachrichten auf einmal

Um das Timeout für die Nachrichtensichtbarkeit für mehrere Nachrichten gleichzeitig festzulegen, erstellen Sie eine Liste von [ChangeMessageVisibilityBatchRequestEntry](#) Objekten, die jeweils eine eindeutige ID-Zeichenfolge und ein Empfangs-Handle enthalten. Übergeben Sie dann die Liste an die Methode der Amazon SQS Client-Klasse. `changeMessageVisibilityBatch`

## Importe

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.ChangeMessageVisibilityBatchRequestEntry;
import java.util.ArrayList;
import java.util.List;
```

## Code

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();

List<ChangeMessageVisibilityBatchRequestEntry> entries =
    new ArrayList<ChangeMessageVisibilityBatchRequestEntry>();

entries.add(new ChangeMessageVisibilityBatchRequestEntry(
    "unique_id_msg1",
    sqs.receiveMessage(queue_url)
        .getMessages()
        .get(0)
        .getReceiptHandle())
    .withVisibilityTimeout(timeout));

entries.add(new ChangeMessageVisibilityBatchRequestEntry(
    "unique_id_msg2",
    sqs.receiveMessage(queue_url)
        .getMessages()
        .get(0)
        .getReceiptHandle())
    .withVisibilityTimeout(timeout + 200));

sqs.changeMessageVisibilityBatch(queue_url, entries);
```

Das [vollständige Beispiel](#) finden Sie unter. [GitHub](#)

## Weitere Infos

- [Visibility Timeout](#) im Amazon SQS Developer Guide
- [SetQueueAttributes](#) in der Amazon SQS API-Referenz
- [GetQueueAttributes](#) in der Amazon SQS API-Referenz
- [ReceiveMessage](#) in der Amazon SQS API-Referenz

- [ChangeMessageVisibility](#) in der Amazon SQS API-Referenz
- [ChangeMessageVisibilityBatch](#) in der Amazon SQS API-Referenz

## Verwenden von Warteschlangen für unzustellbare Briefe in Amazon SQS

Amazon SQS bietet Unterstützung für Warteschlangen mit uneingeschränkten Briefen. Andere (Quell-)Warteschlangen können Nachrichten, die nicht erfolgreich verarbeitet werden konnten, an die Warteschlange für unzustellbare Nachrichten senden. Sie können diese Nachrichten in der Warteschlange für unzustellbare Nachrichten sammeln und isolieren, um zu bestimmen, warum die Verarbeitung fehlgeschlagen ist.

### Erstellen einer Warteschlange für unzustellbare Nachrichten

Eine Warteschlange für unzustellbare Nachrichten wird wie eine reguläre Warteschlange erstellt, hat aber folgende Einschränkungen:

- Eine Warteschlange für unzustellbare Nachrichten muss den gleichen Typ der Warteschlange (FIFO oder Standard) wie die Quell-Warteschlange haben.
- Eine Warteschlange für eingehende Nachrichten muss mit derselben AWS-Konto AND-Region wie die Quellwarteschlange erstellt werden.

Hier erstellen wir zwei identische Amazon SQS Warteschlangen, von denen eine als Warteschlange für eingehende Nachrichten dient:

#### Importe

```
import com.amazonaws.services.sqs.AmazonSQS;  
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;  
import com.amazonaws.services.sqs.model.AmazonSQSException;
```

#### Code

```
final AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();  
  
// Create source queue  
try {  
    sqs.createQueue(src_queue_name);  
} catch (AmazonSQSException e) {  
    if (!e.getErrorCode().equals("QueueAlreadyExists")) {
```

```
        throw e;
    }
}

// Create dead-letter queue
try {
    sqs.createQueue(dl_queue_name);
} catch (AmazonSQSException e) {
    if (!e.getErrorCode().equals("QueueAlreadyExists")) {
        throw e;
    }
}
```

Das [vollständige Beispiel](#) finden Sie unter [GitHub](#)

## Zuweisen einer Warteschlange für unzustellbare Nachrichten an eine Quell-Warteschlange

Sie können eine Warteschlange für unzustellbare Nachrichten zuweisen, indem Sie als Erstes eine Redrive-Richtlinie erstellen und die Richtlinie dann in den Attributen der Warteschlange festlegen. Eine Redrive-Richtlinie wird in JSON angegeben und enthält den ARN der Warteschlange für unzustellbare Nachrichten sowie die maximale Anzahl an Malen, die eine Nachricht empfangen und nicht verarbeitet werden kann, bevor sie an die Warteschlange für unzustellbare Nachrichten gesendet wird.

Um die Redrive-Richtlinie für Ihre Quell-Warteschlange festzulegen, rufen Sie die `setQueueAttributes` Methode der `AmazonSQS`-Klasse mit einem [SetQueueAttributesRequest](#) Objekt auf, für das Sie das `RedrivePolicy` Attribut mit Ihrer JSON-Redrive-Richtlinie festgelegt haben.

### Importe

```
import com.amazonaws.services.sqs.model.GetQueueAttributesRequest;
import com.amazonaws.services.sqs.model.GetQueueAttributesResult;
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;
```

### Code

```
String dl_queue_url = sqs.getQueueUrl(dl_queue_name)
    .getQueueUrl();
```

```
GetQueueAttributesResult queue_attrs = sqs.getQueueAttributes(  
    new GetQueueAttributesRequest(dl_queue_url)  
        .withAttributeNames("QueueArn"));  
  
String dl_queue_arn = queue_attrs.getAttributes().get("QueueArn");  
  
// Set dead letter queue with redrive policy on source queue.  
String src_queue_url = sqs.getQueueUrl(src_queue_name)  
    .getQueueUrl();  
  
SetQueueAttributesRequest request = new SetQueueAttributesRequest()  
    .withQueueUrl(src_queue_url)  
    .addAttributesEntry("RedrivePolicy",  
        "{\n\"maxReceiveCount\":\n\"5\", \n\"deadLetterTargetArn\":\n\""  
        + dl_queue_arn + "\n}");  
  
sqs.setQueueAttributes(request);
```

[Das vollständige Beispiel finden Sie unter. GitHub](#)

## Weitere Infos

- [Verwenden von Amazon SQS Dead Letter Queues](#) im Amazon SQS Entwicklerhandbuch
- [SetQueueAttributes](#) in der Amazon SQS API-Referenz

## Amazon SWF Beispiele für die Verwendung der AWS SDK für Java

[Amazon SWF ist ein Workflow-Management-Service, der Entwicklern hilft, verteilte Workflows zu erstellen und zu skalieren, die parallel oder sequentielle Schritte umfassen können, die aus Aktivitäten, untergeordneten Workflows oder sogar Lambda-Aufgaben bestehen können.](#)

Es gibt zwei Möglichkeiten, mit dem zu arbeiten: Amazon SWF mit dem AWS SDK für JavaSWF-Client-Objekt oder mit dem für Java. AWS Flow Framework Das AWS Flow Framework für Java ist anfangs schwieriger zu konfigurieren, da es häufig Anmerkungen verwendet und auf zusätzliche Bibliotheken wie AspectJ und das Spring Framework angewiesen ist. Bei großen oder komplexen Projekten sparen Sie jedoch Codierungszeit, indem Sie for Java verwenden. AWS Flow Framework Weitere Informationen finden Sie im [AWS Flow Framework for Java Developer Guide](#).

Dieser Abschnitt enthält Beispiele für die direkte Programmierung Amazon SWF mit dem AWS SDK für Java Client.

## Themen

- [SWF-Grundlagen](#)
- [Eine einfache Amazon SWF Anwendung erstellen](#)
- [Lambda Aufgaben](#)
- [Korrektes Herunterfahren von Aktivitäts- und Workflow-Workern](#)
- [Registrieren von Domänen](#)
- [Auflisten von Domänen](#)

## SWF-Grundlagen

Dies sind allgemeine Muster für die Arbeit Amazon SWF mit der AWS SDK für Java. Dies soll hauptsächlich als Referenz dienen. Ein vollständigeres Einführungstutorial finden Sie unter [Erstellen einer einfachen Amazon SWF Anwendung](#).

## Abhängigkeiten

Für Amazon SWF Basisanwendungen sind die folgenden Abhängigkeiten erforderlich, die im Lieferumfang von enthalten sind AWS SDK für Java:

- aws-java-sdk-1.12.\*.jar
- commons-logging-1.2.\*.jar
- httpclient-4.3.\*.jar
- httpcore-4.3.\*.jar
- jackson-annotations-2.12.\*.jar
- jackson-core-2.12.\*.jar
- jackson-databind-2.12.\*.jar
- joda-time-2.8.\*.jar

### Note

Die Versionsnummern dieser Pakete hängen von der Version des SDK ab, die Sie haben, aber die Versionen, die mit dem SDK geliefert werden, wurden auf Kompatibilität getestet und sollten Sie verwenden.

AWS Flow Framework für Java-Anwendungen sind zusätzliche Einstellungen und zusätzliche Abhängigkeiten erforderlich. Weitere Informationen [AWS Flow Framework zur Verwendung des Frameworks finden Sie im for Java Developer Guide](#).

## Importe

Im Allgemeinen können Sie die folgenden Importe für Code-Entwicklung nutzen:

```
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.*;
```

Sie sollten jedoch nur die Klassen importieren, die Sie wirklich benötigen.

Dazu geben Sie wahrscheinlich bestimmte Klassen im Workspace `com.amazonaws.services.simpleworkflow.model` an:

```
import com.amazonaws.services.simpleworkflow.model.PollForActivityTaskRequest;
import com.amazonaws.services.simpleworkflow.model.RespondActivityTaskCompletedRequest;
import com.amazonaws.services.simpleworkflow.model.RespondActivityTaskFailedRequest;
import com.amazonaws.services.simpleworkflow.model.TaskList;
```

Wenn Sie das AWS Flow Framework für Java verwenden, importieren Sie Klassen aus dem `com.amazonaws.services.simpleworkflow.flow` Workspace. Zum Beispiel:

```
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
import com.amazonaws.services.simpleworkflow.flow.ActivityWorker;
```

### Note

AWS Flow Framework Für Java gelten zusätzliche Anforderungen, die über die Basisversion hinausgehen AWS SDK für Java. Weitere Informationen finden Sie im [AWS Flow Framework for Java Developer Guide](#).

## Verwenden der SWF-Client-Klasse

Ihre grundlegende Schnittstelle zu Amazon SWF sind entweder die [AmazonSimpleWorkflowAsyncClient](#) Klassen [AmazonSimpleWorkflowClient](#) oder. Der Unterschied zwischen diesen Klassen besteht darin, dass die `*AsyncClient`-Klasse [Future](#)-Objekte für gleichzeitige (asynchrone) Programmierung zurückgibt.

```
AmazonSimpleWorkflowClient swf = AmazonSimpleWorkflowClientBuilder.defaultClient();
```

## Eine einfache Amazon SWF Anwendung erstellen

In diesem Thema werden Sie in die Programmierung von [Amazon SWF](#) Anwendungen mit dem AWS SDK für Java eingeführt und dabei einige wichtige Konzepte vorgestellt.

### Über das Beispiel

Das Beispielprojekt erstellt einen Workflow mit einer einzigen Aktivität, der Workflow-Daten akzeptiert, die über die AWS Cloud übertragen werden (in der HelloWorld Tradition ist dies der Name einer zu begrüßenden Person) und anschließend eine Begrüßung als Antwort ausgibt.

Obwohl dies auf den ersten Blick sehr einfach erscheint, bestehen Amazon SWF Anwendungen aus einer Reihe von Teilen, die zusammenarbeiten:

- Einer Domäne als logischem Container für die Ausführungsdaten des Workflows.
- Einem oder mehreren Workflows, die Code-Komponenten darstellen, mit denen die logische Reihenfolge der Ausführung für die Aktivitäten und untergeordneten Workflows Ihres Workflows definiert wird.
- Einem Workflow-Worker, auch Entscheider genannt, der Abfragen für Entscheidungsaufgaben ausführt und daraufhin Aktivitäten oder untergeordnete Workflows plant.
- Einer oder mehreren Aktivitäten, die jeweils eine Arbeitseinheit im Workflow darstellen.
- Einem Aktivitäts-Worker, der Abfragen für Aktivitätsaufgaben durchführt und als Reaktion Aktivitätsmethoden ausführt.
- Eine oder mehrere Aufgabenlisten, bei denen es sich um Warteschlangen handelt, die Amazon SWF dazu dienen, Anfragen an die Workflow- und Aktivitätsmitarbeiter zu richten. Aufgaben in einer Aufgabenliste für Workflow-Worker werden Entscheidungsaufgaben genannt. Aufgaben für Aktivitäts-Worker nennen sich Aktivitätsaufgaben.
- Einem Workflow-Starter, der mit der Ausführung des Workflows beginnt.

Amazon SWF Orchestriert hinter den Kulissen den Betrieb dieser Komponenten, koordiniert ihren Fluss aus der AWS Cloud, leitet Daten zwischen ihnen weiter, verarbeitet Timeouts und Heartbeat-Benachrichtigungen und protokolliert den Verlauf der Workflow-Ausführung.

## Voraussetzungen

### Entwicklungsumgebung

Die Entwicklungsumgebung in dieser Anleitung besteht aus:

- Das Tool [AWS SDK für Java](#).
- [Apache Maven](#) (3.3.1).
- JDK 1.7 oder neuer. Diese Anleitung wurde mit JDK 1.8.0 entwickelt und getestet.
- Einen guten Java-Texteditor (Ihrer Wahl).

#### Note

Wenn Sie ein anderes Build-System als Maven verwenden, können Sie trotzdem ein Projekt mit den entsprechenden Schritten für Ihre Umgebung erstellen und dabei die hier bereitgestellten Konzepte verwenden. Weitere Informationen zur Konfiguration und Verwendung von AWS SDK für Java mit den verschiedenen Build-Systemen finden Sie unter [Erste Schritte](#).

Ebenso, aber mit größerem Aufwand, können die hier gezeigten Schritte mit jedem der AWS SDKs mit Unterstützung für implementiert werden Amazon SWF.

Alle erforderlichen externen Abhängigkeiten sind im Lieferumfang von enthalten AWS SDK für Java, sodass Sie nichts zusätzlich herunterladen müssen.

### AWS Zugriff

Um dieses Tutorial erfolgreich durcharbeiten zu können, benötigen Sie Zugriff auf das AWS Zugangsportal, wie [im Abschnitt zur Grundkonfiguration dieses Handbuchs beschrieben](#).

In den Anweisungen wird beschrieben, wie Sie auf temporäre Anmeldeinformationen zugreifen, die Sie kopieren und in Ihre lokale gemeinsam genutzte `credentials` Datei einfügen. Die temporären Anmeldeinformationen, die Sie einfügen, müssen einer IAM-Rolle zugeordnet sein AWS IAM Identity Center , die über Zugriffsberechtigungen für Amazon SWF verfügt. Nach dem Einfügen der temporären Anmeldeinformationen sieht Ihre `credentials` Datei wie folgt aus.

```
[default]
```



3. Achten Sie darauf, dass Maven das Projekt mit Unterstützung für JDK 1.7+ erstellt. Fügen Sie Folgendes in der Datei `<dependencies>` zu Ihrem Projekt hinzu (vor oder nach dem Block `pom.xml`):

```
<build>
  <plugins>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.6.1</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>
  </plugins>
</build>
```

## Entwickeln des Projekts

Das Beispielprojekt umfasst vier separate Anwendungen, die wir einzeln untersuchen:

- `HelloTypes.java` — enthält die Domänen-, Aktivitäts- und Workflow-Typdaten des Projekts, die mit den anderen Komponenten gemeinsam genutzt werden. Außerdem übernimmt diese Datei das Registrieren dieser Typen mit SWF.
- `ActivityWorker.java` --enthält den Activity Worker, der nach Aktivitätsaufgaben fragt und daraufhin Aktivitäten ausführt.
- `WorkflowWorker.java` — enthält den Workflow-Worker (Decider), der Entscheidungsaufgaben abfragt und neue Aktivitäten plant.
- `WorkflowStarter.java` --enthält den Workflow-Starter, der eine neue Workflow-Ausführung startet, wodurch SWF beginnt, Entscheidungs- und Workflow-Aufgaben zu generieren, die Ihre Worker bearbeiten können.

### Allgemeine Schritte für alle Quelldateien

Alle Dateien, die Sie erstellen, um Ihre Java-Klassen zu integrieren, haben ein paar Dinge gemeinsam. Aus zeitlichen Gründen werden die folgenden Schritte jedes Mal implizit vorausgesetzt, wenn Sie eine neue Datei zum Projekt hinzufügen:

1. Erstellen Sie die Datei im Verzeichnis `src/main/java/aws/example/helloswf/` des Projekts.
2. Fügen Sie eine `package`-Deklaration am Anfang jeder Datei hinzu, um ihren Namespace zu deklarieren. Das Beispielprojekt nutzt:

```
package aws.example.helloswf;
```

3. Fügen Sie `import` Deklarationen für die [AmazonSimpleWorkflowClient](#)-Klasse und für mehrere Klassen im `com.amazonaws.services.simpleworkflow.model` Namespace hinzu. Der Einfachheit halber verwenden wir:

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.*;
```

## Registrieren von Domäne und Workflow- und Aktivitätstypen

Als Erstes erstellen wir eine neue ausführbare Klasse, `HelloTypes.java`. Diese Datei enthält freigegebene Daten, die verschiedenen Teilen Ihres Workflows bekannt sein müssen, z. B. die Namen und Version Ihrer Aktivitäten und Workflow-Typen, den Namen der Domäne und den Namen der Aufgabenliste.

1. Öffnen Sie den Texteditor und erstellen Sie die Datei `HelloTypes.java`. Fügen Sie eine `Package`-Deklaration und die Importe laut den [allgemeinen Schritten](#) hinzu.
2. Deklarieren Sie die `HelloTypes`-Klasse und geben Sie Werte für Ihre registrierten Aktivitäts- und Workflow-Typen an:

```
public static final String DOMAIN = "HelloDomain";
public static final String TASKLIST = "HelloTasklist";
public static final String WORKFLOW = "HelloWorkflow";
public static final String WORKFLOW_VERSION = "1.0";
public static final String ACTIVITY = "HelloActivity";
public static final String ACTIVITY_VERSION = "1.0";
```

Diese Werte werden im gesamten Code verwendet.

- Erstellen Sie nach den String-Deklarationen eine Instanz der [AmazonSimpleWorkflowClient](#)-Klasse. Dies ist die grundlegende Schnittstelle zu den Amazon SWF Methoden, die von der bereitgestellt werden AWS SDK für Java.

```
private static final AmazonSimpleWorkflow swf =  
  
    AmazonSimpleWorkflowClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
```

Im vorherigen Codeausschnitt wird davon ausgegangen, dass dem Profil temporäre Anmeldeinformationen zugeordnet sind. `default` Wenn Sie ein anderes Profil verwenden, ändern Sie den obigen Code wie folgt und `profile_name` ersetzen Sie ihn durch den Namen des tatsächlichen Profilnamens.

```
private static final AmazonSimpleWorkflow swf =  
    AmazonSimpleWorkflowClientBuilder  
        .standard()  
        .withCredentials(new ProfileCredentialsProvider("profile_name"))  
        .withRegion(Regions.DEFAULT_REGION)  
        .build();
```

- Fügen Sie eine neue Funktion für die Registrierung einer SWF-Domäne hinzu. Bei einer Domäne handelt es sich um einen logischen Container für eine Reihe von zugehörigen SWF-Aktivitäten und Workflow-Typen. SWF-Komponenten können nur miteinander kommunizieren, wenn sie in derselben Domäne sind.

```
try {  
    System.out.println("** Registering the domain '" + DOMAIN + "'.");  
    swf.registerDomain(new RegisterDomainRequest()  
        .withName(DOMAIN)  
        .withWorkflowExecutionRetentionPeriodInDays("1"));  
} catch (DomainAlreadyExistsException e) {  
    System.out.println("** Domain already exists!");  
}
```

Wenn Sie eine Domain registrieren, geben Sie ihr einen Namen (eine beliebige Gruppe von 1 bis 256 Zeichen mit Ausnahme von: `.,/|`, Steuerzeichen oder der wörtlichen Zeichenfolge `"arn"`) und eine Aufbewahrungsfrist, d. h. die Anzahl der Tage, in denen die Ausführungsverlaufsdaten Ihres Workflows nach Abschluss einer Workflow-Ausführung aufbewahrt Amazon SWF werden.

Der maximale Aufbewahrungszeitraum für die Workflow-Ausführung ist 90 Tage. Weitere Informationen finden Sie unter [RegisterDomainRequest](#).

Wenn eine Domäne mit diesem Namen bereits existiert, [DomainAlreadyExistsException](#) wird aktiviert. Da uns nicht interessiert, ob die Domäne schon erstellt wurde, können wir die Ausnahme ignorieren.

#### Note

Dieser Code veranschaulicht ein allgemeines Muster bei der Arbeit mit AWS SDK für Java Methoden: Daten für die Methode werden von einer Klasse im `simpleworkflow.model` Namespace bereitgestellt, die Sie mithilfe der verkettbaren Methoden instanziiieren und auffüllen. `0with*`

5. Fügen Sie eine neue Funktion für die Registrierung eines neuen Aktivitätstyps hinzu. Eine Aktivität stellt eine Arbeitseinheit in Ihrem Workflow dar.

```
try {
    System.out.println("** Registering the activity type '" + ACTIVITY +
        "-" + ACTIVITY_VERSION + "'.");
    swf.registerActivityType(new RegisterActivityTypeRequest()
        .withDomain(DOMAIN)
        .withName(ACTIVITY)
        .withVersion(ACTIVITY_VERSION)
        .withDefaultTaskList(new TaskList().withName(TASKLIST))
        .withDefaultTaskScheduleToStartTimeout("30")
        .withDefaultTaskStartToCloseTimeout("600")
        .withDefaultTaskScheduleToCloseTimeout("630")
        .withDefaultTaskHeartbeatTimeout("10"));
} catch (TypeAlreadyExistsException e) {
    System.out.println("** Activity type already exists!");
}
```

Ein Aktivitätstyp wird durch einen Namen und eine Version angegeben, die zum Unterscheiden der Aktivitäten von denen anderen Dateien in der Domäne, in der sie registriert sind, verwendet werden. Aktivitäten enthalten außerdem eine Reihe von optionalen Parametern, wie die Standard-Aufgabenliste für den Empfang von Aufgaben und Daten aus SWF und eine Reihe verschiedener Timeouts, mit denen Sie Einschränkungen dafür, wie lange verschiedene Teile der Aktivität ausgeführt werden dürfen, festlegen können. Weitere Informationen finden Sie unter [RegisterActivityTypeRequest](#).

**Note**

Alle Timeout-Werte werden in Sekunden angegeben. Eine vollständige Beschreibung der Auswirkungen von [Amazon SWF Timeouts auf Ihre Workflow-Ausführungen finden Sie unter Timeout-Typen](#).

Wenn der Aktivitätstyp, den Sie registrieren möchten, bereits existiert, [TypeAlreadyExistsException](#) wird ein ausgelöst. Fügen Sie eine neue Funktion für die Registrierung eines neuen Workflow-Typs hinzu. Ein Workflow, auch Entscheider genannt, stellt die Logik Ihrer Workflow-Ausführung dar.

+

```
try {
    System.out.println("*** Registering the workflow type '" + WORKFLOW +
        "-" + WORKFLOW_VERSION + "'.");
    swf.registerWorkflowType(new RegisterWorkflowTypeRequest()
        .withDomain(DOMAIN)
        .withName(WORKFLOW)
        .withVersion(WORKFLOW_VERSION)
        .withDefaultChildPolicy(ChildPolicy.TERMINATE)
        .withDefaultTaskList(new TaskList().withName(TASKLIST))
        .withDefaultTaskStartToCloseTimeout("30"));
} catch (TypeAlreadyExistsException e) {
    System.out.println("*** Workflow type already exists!");
}
```

+

Ähnlich wie bei Aktivitätstypen werden Workflow-Typen durch einen Namen und eine Version identifiziert und haben auch konfigurierbare Timeouts. Weitere Informationen finden Sie unter [RegisterWorkflowTypeRequest](#).

+

Wenn der Workflowtyp, den Sie registrieren möchten, bereits existiert, [TypeAlreadyExistsException](#) wird ein ausgelöst. Markieren Sie die Klasse schließlich als ausführbar, indem Sie eine `main`-Methode hinzufügen. Diese registriert die Domäne, den Aktivitätstyp sowie den Workflow-Typ:

+

```
registerDomain();
registerWorkflowType();
registerActivityType();
```

Jetzt können Sie die Anwendung [erstellen](#) und [ausführen](#), um das Registrierungsskript auszuführen. Sie können aber auch mit dem Entwickeln der Aktivitäts- und Workflow-Worker fortfahren. Sobald die Domäne, der Workflow und die Aktivität registriert wurden, müssen Sie sie nicht erneut ausführen. Diese Typen bleiben bestehen, bis Sie sie selbst als veraltet kennzeichnen.

### Implementieren des Aktivitäts-Workers

Eine Aktivität ist die grundlegende Arbeitseinheit in einem Workflow. Ein Workflow stellt die Logik bereit und plant auszuführende Aktivitäten (oder andere Aktionen) als Reaktion auf Entscheidungsaufgaben. Ein typischer Workflow besteht normalerweise aus einer Reihe von Aktivitäten, die synchron, asynchron oder gemischt ausgeführt werden können.

Der Activity Worker ist der Code, der nach Aktivitätsaufgaben fragt, die von Amazon SWF als Reaktion auf Workflow-Entscheidungen generiert werden. Wird eine Aktivitätsaufgabe empfangen, wird die zugehörige Aktivität ausgeführt und eine Erfolg-/Fehlermeldung an den Workflow zurückgegeben.

Wir implementieren einen einfachen Aktivitäts-Worker, der eine einzelne Aktivität ausführt.

1. Öffnen Sie den Texteditor und erstellen Sie die Datei `ActivityWorker.java`. Fügen Sie eine Package-Deklaration und die Importe laut den [allgemeinen Schritten](#) hinzu.

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.*;
```

2. Fügen Sie die `ActivityWorker` Klasse der Datei hinzu und geben Sie ihr ein Datenelement für einen SWF-Client, mit dem Amazon SWF wir interagieren werden:

```
private static final AmazonSimpleWorkflow swf =

AmazonSimpleWorkflowClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
```

3. Fügen Sie die Methode, die wir nutzen werden, als Aktivität hinzu:

```
private static String sayHello(String input) throws Throwable {
    return "Hello, " + input + "!";
}
```

Die Aktivität nimmt einfach eine Zeichenfolge entgegen, kombiniert sie zu einer Begrüßung und gibt das Ergebnis zurück. Es ist zwar unwahrscheinlich, dass diese Aktivität eine Ausnahme auslöst. Dennoch empfiehlt es sich, Aktivitäten zu entwerfen, die einen Fehler auslösen können, wenn ein Fehler auftritt.

4. Fügen Sie eine `main`-Methode hinzu. Wir verwenden sie als Abfragemethode der Aktivitätsaufgabe. Wir starten sie, indem wir Code hinzufügen, der die Aufgabenliste nach Aktivitätsaufgaben abfragt:

```
System.out.println("Polling for an activity task from the tasklist '"
    + HelloTypes.TASKLIST + "' in the domain '" +
    HelloTypes.DOMAIN + "'.");

ActivityTask task = swf.pollForActivityTask(
    new PollForActivityTaskRequest()
        .withDomain(HelloTypes.DOMAIN)
        .withTaskList(
            new TaskList().withName(HelloTypes.TASKLIST)));

String task_token = task.getTaskToken();
```

Die Aktivität empfängt Aufgaben von, Amazon SWF indem sie die `pollForActivityTask` Methode des SWF-Clients aufruft und dabei die Domäne und die Aufgabenliste angibt, die in der übergebenen Datei verwendet werden sollen. [PollForActivityTaskRequest](#)

Sobald eine Aufgabe empfangen wird, rufen wir eine eindeutige Kennung für sie ab, indem wir die `getTaskToken`-Methode der Aufgabe aufrufen.

5. Schreiben Sie als Nächstes Code zum Verarbeiten der eingehenden Aufgaben. Fügen Sie Folgendes zur `main`-Methode hinzu, und zwar direkt nach dem Code, der die Aufgabe abrufen und deren Aufgabentoken ermittelt.

```
if (task_token != null) {
    String result = null;
    Throwable error = null;
```

```
try {
    System.out.println("Executing the activity task with input '" +
        task.getInput() + "'.");
    result = sayHello(task.getInput());
} catch (Throwable th) {
    error = th;
}

if (error == null) {
    System.out.println("The activity task succeeded with result '"
        + result + "'.");
    swf.respondActivityTaskCompleted(
        new RespondActivityTaskCompletedRequest()
            .withTaskToken(task_token)
            .withResult(result));
} else {
    System.out.println("The activity task failed with the error '"
        + error.getClass().getSimpleName() + "'.");
    swf.respondActivityTaskFailed(
        new RespondActivityTaskFailedRequest()
            .withTaskToken(task_token)
            .withReason(error.getClass().getSimpleName())
            .withDetails(error.getMessage()));
}
}
```

Wenn das Aufgabentoken ungleich null ist, beginnen wir die Ausführung der Aktivitätsmethode (sayHello) und übergeben dabei die Eingabedaten, die mit der Aufgabe mitgesendet wurden.

Wenn die Aufgabe erfolgreich war (es wurde kein Fehler generiert), reagiert der Worker auf SWF, indem er die `respondActivityTaskCompleted` Methode des SWF-Clients mit einem [RespondActivityTaskCompletedRequest](#) Objekt aufruft, das das Task-Token und die Ergebnisdaten der Aktivität enthält.

Wenn die Aufgabe dagegen fehlgeschlagen ist, antworten wir, indem wir die `respondActivityTaskFailed` Methode mit einem [RespondActivityTaskFailedRequest](#) Objekt aufrufen und ihr das Task-Token und Informationen über den Fehler übergeben.

**Note**

Diese Aktivität wird nicht korrekt beendet, wenn sie unsanft abgebrochen wird. Dies geht zwar über die Grenzen dieser Anleitung hinaus, doch eine alternative Implementierung dieses Aktivitäts-Workers finden Sie im begleitenden Thema [Korrektes Herunterfahren von Aktivitäts- und Workflow-Workern](#).

## Implementieren des Workflow-Workers

Die Workflow-Logik liegt in einem Codeteil, der Workflow-Worker genannt wird. Der Workflow-Worker fragt nach Entscheidungsaufgaben ab, die Amazon SWF in der Domäne und in der Standard-Taskliste gesendet wurden, für die der Workflowtyp registriert wurde.

Wenn der Workflow-Worker eine Aufgabe erhält, wird eine Art Entscheidung gefällt (in der Regel, ob eine neue Aktivität geplant werden soll oder nicht) und eine entsprechende Aktion ausgeführt (z. B. zur Planung der Aktivität).

1. Öffnen Sie den Texteditor und erstellen Sie die Datei `WorkflowWorker.java`. Fügen Sie eine Package-Deklaration und die Importe laut den [allgemeinen Schritten](#) hinzu.
2. Fügen Sie einige zusätzliche Importe in die Datei ein:

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.*;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
```

3. Deklarieren Sie die `WorkflowWorker` Klasse und erstellen Sie eine Instanz der [AmazonSimpleWorkflowClient](#) Klasse, die für den Zugriff auf SWF-Methoden verwendet wird.

```
private static final AmazonSimpleWorkflow swf =
    AmazonSimpleWorkflowClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
```

4. Fügen Sie die `main`-Methode hinzu. Die Methode wird in Schleife ausgeführt und ruft Entscheidungsaufgaben mit der `pollForDecisionTask`-Methode des SWF-Clients ab. Die [PollForDecisionTaskRequest](#) stellt die Details bereit.

```
    PollForDecisionTaskRequest task_request =
        new PollForDecisionTaskRequest()
            .withDomain>HelloTypes.DOMAIN)
            .withTaskList(new TaskList().withName>HelloTypes.TASKLIST));

    while (true) {
        System.out.println(
            "Polling for a decision task from the tasklist '" +
            HelloTypes.TASKLIST + "' in the domain '" +
            HelloTypes.DOMAIN + "'.");

        DecisionTask task = swf.pollForDecisionTask(task_request);

        String taskToken = task.getTaskToken();
        if (taskToken != null) {
            try {
                executeDecisionTask(taskToken, task.getEvents());
            } catch (Throwable th) {
                th.printStackTrace();
            }
        }
    }
}
```

Sobald eine Aufgabe empfangen wird, rufen wir ihre `getTaskToken`-Methode auf. Diese gibt eine Zeichenfolge zur Erkennung der Aufgabe zurück. Wenn das zurückgegebene Token nicht vorhanden ist `null`, verarbeiten wir es in der `executeDecisionTask` Methode weiter und übergeben ihm das Task-Token und die Liste der mit der Aufgabe gesendeten [HistoryEvent](#) Objekte.

5. Fügen Sie die `executeDecisionTask`-Methode hinzu. Sie nimmt das Aufgabentoken (einen `String`) und eine `HistoryEvent`-Liste entgegen.

```
List<Decision> decisions = new ArrayList<Decision>();
String workflow_input = null;
int scheduled_activities = 0;
int open_activities = 0;
boolean activity_completed = false;
String result = null;
```

Wir richten auch einige Datenmitglieder zur Nachverfolgung ein, u. a.:

- Eine Liste mit [Decision](#)-Objekten, mit denen die Ergebnisse der Aufgabenverarbeitung berichtet werden.
  - Eine Zeichenfolge für die Workflow-Eingabe, die durch das Ereignis "WorkflowExecutionStarted" bereitgestellt wird
  - Eine Zählung der geplanten und offenen (aktiven) Aktivitäten. So wird die Planung von Aktivitäten vermieden, die bereits geplant wurden oder momentan ausgeführt werden.
  - Einen boolescher Wert, der angibt, ob die Aktivität abgeschlossen ist.
  - Eine Zeichenfolge, die die Aktivitätsergebnisse für die Rückgabe als unser Workflow-Ergebnis speichert.
6. Fügen Sie als Nächstes Code in die `executeDecisionTask`-Methode ein, der die mit der Aufgabe mitgesendeten `HistoryEvent`-Objekte verarbeitet, je nachdem, welcher Ereignistyp von der `getEventType`-Methode gemeldet wurde.

```
System.out.println("Executing the decision task for the history events: [");
for (HistoryEvent event : events) {
    System.out.println(" " + event);
    switch(event.getEventType()) {
        case "WorkflowExecutionStarted":
            workflow_input =
                event.getWorkflowExecutionStartedEventAttributes()
                    .getInput();
            break;
        case "ActivityTaskScheduled":
            scheduled_activities++;
            break;
        case "ScheduleActivityTaskFailed":
            scheduled_activities--;
            break;
        case "ActivityTaskStarted":
            scheduled_activities--;
            open_activities++;
            break;
        case "ActivityTaskCompleted":
            open_activities--;
            activity_completed = true;
            result = event.getActivityTaskCompletedEventAttributes()
                .getResult();
            break;
        case "ActivityTaskFailed":
            open_activities--;
```

```

        break;
    case "ActivityTaskTimedOut":
        open_activities--;
        break;
    }
}
System.out.println("]");

```

Für die Zwecke unseres Workflows interessieren wir uns am meisten für:

- das Ereignis `WorkflowExecutionStarted` "", das angibt, dass die Workflow-Ausführung gestartet wurde (was normalerweise bedeutet, dass Sie die erste Aktivität im Workflow ausführen sollten), und das die erste Eingabe für den Workflow bereitstellt. In diesem Fall handelt es sich um den Namen für unsere Begrüßung. Deswegen speichern wir die Daten in einer Zeichenfolge, während wir die auszuführende Aktivität planen.
- das Ereignis `ActivityTaskCompleted` "", das gesendet wird, sobald die geplante Aktivität abgeschlossen ist. Die Ereignisdaten enthalten auch den Rückgabewert der abgeschlossenen Aktivität. Da wir nur eine Aktivität haben, verwenden wir diesen Wert als Ergebnis des gesamten Workflows.

Die anderen Ereignistypen können verwendet werden, wenn Ihre Workflows es erfordern.

Informationen zu den einzelnen Ereignistypen finden Sie in der [HistoryEvent](#)Kursbeschreibung.

+ HINWEIS: Zeichenketten in `switch` Anweisungen wurden in Java 7 eingeführt. Wenn Sie eine frühere Version von Java verwenden, können Sie die [EventType](#)Klasse verwenden, um den von `String history_event.getType()` zurückgegebenen Wert in einen Enum-Wert und dann, `String` falls erforderlich, wieder in einen umzuwandeln:

```
EventType et = EventType.fromValue(event.getEventType());
```

1. Fügen Sie nach der `switch`-Anweisung weiteren Code hinzu, um mit einer passenden Entscheidung auf die empfangene Aufgabe zu reagieren.

```

if (activity_completed) {
    decisions.add(
        new Decision()
            .withDecisionType(DecisionType.CompleteWorkflowExecution)
            .withCompleteWorkflowExecutionDecisionAttributes(
                new CompleteWorkflowExecutionDecisionAttributes()

```

```

        .withResult(result));
    } else {
        if (open_activities == 0 && scheduled_activities == 0) {

            ScheduleActivityTaskDecisionAttributes attrs =
                new ScheduleActivityTaskDecisionAttributes()
                    .withActivityType(new ActivityType()
                        .withName(HelloTypes.ACTIVITY)
                        .withVersion(HelloTypes.ACTIVITY_VERSION))
                    .withActivityId(UUID.randomUUID().toString())
                    .withInput(workflow_input);

            decisions.add(
                new Decision()
                    .withDecisionType(DecisionType.ScheduleActivityTask)
                    .withScheduleActivityTaskDecisionAttributes(attrs));
        } else {
            // an instance of HelloActivity is already scheduled or running. Do nothing,
            another
            // task will be scheduled once the activity completes, fails or times out
        }
    }
}

System.out.println("Exiting the decision task with the decisions " + decisions);

```

- Wenn die Aktivität noch nicht geplant wurde, antworten wir mit einer `ScheduleActivityTask` Entscheidung, die Informationen in einer [ScheduleActivityTaskDecisionAttributes](#) Struktur über die Aktivität bereitstellt, die als Nächstes geplant Amazon SWF werden soll, einschließlich aller Daten, die an die Aktivität gesendet Amazon SWF werden sollen.
- Wenn die Aktivität abgeschlossen wurde, betrachten wir den gesamten Workflow als abgeschlossen und antworten mit einer `CompletedWorkflowExecution` Entscheidung, indem wir eine [CompleteWorkflowExecutionDecisionAttributes](#) Struktur ausfüllen, um Einzelheiten über den abgeschlossenen Workflow bereitzustellen. In diesem Fall geben wir das Ergebnis der Aktivität zurück.

In beiden Fällen werden die Entscheidungsinformationen zur `Decision`-Liste hinzugefügt, die oben in der Methode deklariert wurde.

2. Vervollständigen Sie die Entscheidungsaufgabe, indem Sie die Liste der `Decision`-Objekte zurückgeben, die bei der Verarbeitung der Aufgabe erfasst wurden. Fügen Sie den Code am Ende der `executeDecisionTask`-Methode hinzu, die wir schreiben:

```
swf.respondDecisionTaskCompleted(  
    new RespondDecisionTaskCompletedRequest()  
        .withTaskToken(taskToken)  
        .withDecisions(decisions));
```

Die `respondDecisionTaskCompleted`-Methode des SWF-Clients nimmt das Aufgabentoken zur Erkennung der Aufgabe sowie die Liste der Decision-Objekte entgegen.

## Implementieren des Workflow-Starters

Schließlich erstellen wir Code zum Starten der Workflow-Ausführung.

1. Öffnen Sie den Texteditor und erstellen Sie die Datei `WorkflowStarter.java`. Fügen Sie eine Package-Deklaration und die Importe laut den [allgemeinen Schritten](#) hinzu.
2. Fügen Sie die `WorkflowStarter`-Klasse hinzu:

```
package aws.example.helloswf;  
  
import com.amazonaws.regions.Regions;  
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;  
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;  
import com.amazonaws.services.simpleworkflow.model.*;  
  
public class WorkflowStarter {  
    private static final AmazonSimpleWorkflow swf =  
  
    AmazonSimpleWorkflowClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();  
    public static final String WORKFLOW_EXECUTION = "HelloWorldWorkflowExecution";  
  
    public static void main(String[] args) {  
        String workflow_input = "{SWF}";  
        if (args.length > 0) {  
            workflow_input = args[0];  
        }  
  
        System.out.println("Starting the workflow execution '" + WORKFLOW_EXECUTION +  
            "' with input '" + workflow_input + "'.");  
  
        WorkflowType wf_type = new WorkflowType()
```

```
.withName>HelloTypes.WORKFLOW)
.withVersion>HelloTypes.WORKFLOW_VERSION);

Run run = swf.startWorkflowExecution(new StartWorkflowExecutionRequest()
    .withDomain>HelloTypes.DOMAIN)
    .withWorkflowType>wf_type)
    .withWorkflowId>WORKFLOW_EXECUTION)
    .withInput>workflow_input)
    .withExecutionStartToCloseTimeout>"90");

System.out.println("Workflow execution started with the run id '" +
    run.getRunId() + "'.");
}
}
```

Die `WorkflowStarter`-Klasse besteht aus einer einzelnen Methode `main`, die ein optionales Argument entgegen nimmt. Dieses wird auf der Befehlszeile als Eingabedaten für den Workflow übergeben.

Die SWF-Client-Methode `startWorkflowExecution`, verwendet ein [StartWorkflowExecutionRequest](#) Objekt als Eingabe. Zusätzlich zur Angabe der Domäne und des auszuführenden Workflow-Typs geben wir hier Folgendes an:

- einen lesbaren Namen für die Workflow-Ausführung,
- Workflow-Eingabedaten (in unserem Beispiel auf der Befehlszeile angegeben) sowie
- einen Timeout-Wert, der in Sekunden angibt, wie lange die Ausführung des gesamten Workflows dauern darf.

Das [zurückgegebene startWorkflowExecution Run-Objekt](#) stellt eine Run-ID bereit, einen Wert, der verwendet werden kann, um diese bestimmte Workflow-Ausführung in Amazon SWF der Historie Ihrer Workflow-Ausführungen zu identifizieren.

+ HINWEIS: Die Lauf-ID wird von dem Namen der Workflow-Ausführung generiert Amazon SWF, den Sie beim Start der Workflow-Ausführung übergeben, und ist nicht identisch mit diesem.

## Erstellen des Beispiels

Sie können das Beispielprojekt mit Maven erstellen, indem Sie zum `helloswf`-Verzeichnis wechseln und Folgendes eingeben:

```
mvn package
```

Die resultierende `helloswf-1.0.jar`-Datei wird im `target`-Verzeichnis erstellt.

## Ausführen des Beispiels

Das Beispiel besteht aus vier separaten ausführbaren Klassen, die unabhängig voneinander ausgeführt werden.

### Note

Wenn Sie ein Linux-, MacOS- oder Unix-System verwenden, können Sie alle nacheinander in einem einzigen Terminalfenster ausführen. Wenn Sie Windows verwenden, sollten Sie zwei weitere Instances der Eingabeaufforderung öffnen und in jedem Fenster zum `helloswf`-Verzeichnis wechseln.

## Festlegen des Java-Klassenpfads

Obwohl Maven die Abhängigkeiten für Sie erledigt hat, müssen Sie zur Ausführung des AWS Beispiels die SDK-Bibliothek und ihre Abhängigkeiten in Ihrem Java-Klassenpfad bereitstellen. Sie können die `CLASSPATH` Umgebungsvariable entweder auf den Speicherort Ihrer AWS SDK-Bibliotheken und das `third-party/lib` Verzeichnis im SDK setzen, das die erforderlichen Abhängigkeiten enthält:

```
export CLASSPATH='target/helloswf-1.0.jar:/path/to/sdk/lib/*:/path/to/sdk/third-party/lib/*'  
java example.swf.hello.HelloTypes
```

oder verwenden Sie die `-cp` Option des `java` Befehls, um den Klassenpfad festzulegen, während die einzelnen Anwendungen ausgeführt werden.

```
java -cp target/helloswf-1.0.jar:/path/to/sdk/lib/*:/path/to/sdk/third-party/lib/* \  
example.swf.hello.HelloTypes
```

Welche Version Sie bevorzugen, liegt an Ihnen. Wenn Sie keine Probleme beim Erstellen des Codes hatten, versuchen beide, die Beispiele auszuführen, und es wird eine Reihe von "NoClassDefFound"-Fehlern angezeigt. Dies liegt wahrscheinlich daran, dass der Klassenpfad falsch gesetzt ist.

## Registrieren von Domäne und Workflow- und Aktivitätstypen

Vor der Ausführung Ihrer Worker und des Workflow-Starters müssen Sie die Domäne und die Workflow- und Aktivitätstypen registrieren. Der entsprechende Code wurde in den [Arbeitsablauf „Eine Domäne registrieren“](#) und in den Aktivitätstypen implementiert.

Nach dem Erstellen und [Festlegen des CLASSPATH](#) können Sie den Code zur Registrierung mit folgendem Befehl ausführen:

```
echo 'Supply the name of one of the example classes as an argument.'
```

## Starten der Aktivitäts- und Workflow-Worker

Nachdem die Typen nun registriert sind, können Sie die Aktivitäts- und Workflow-Worker starten. Diese werden weiterhin ausgeführt und nach Aufgaben abgefragt, bis sie beendet werden. Sie sollten sie also entweder in separaten Terminalfenstern ausführen, oder, wenn Sie unter Linux, macOS oder Unix arbeiten, können Sie den & Operator verwenden, um zu veranlassen, dass jeder von ihnen bei der Ausführung einen eigenen Prozess erzeugt.

```
echo 'If there are arguments to the class, put them in quotes after the class  
name.'  
exit 1
```

Wenn Sie diese Befehle in separaten Fenstern laufen lassen, lassen Sie den letzten &-Operator in jeder Zeile weg.

## Starten der Workflow-Ausführung

Nachdem die Aktivitäts- und Workflow-Worker nun Abfragen durchführen, können Sie die Workflow-Ausführung starten. Dieser Prozess läuft so lange, bis der Workflow den Status "abgeschlossen" zurückgibt. Führen Sie ihn in einem neuen Terminal-Fenster aus (außer Sie haben die Worker mit dem &-Operator in ihre eigenen separaten Prozesse abzweigen lassen).

```
fi
```

### Note

Wenn Sie eigene Eingabedaten angeben möchten, die zuerst an den Workflow und dann an die Aktivität übergeben werden, fügen Sie sie zur Befehlszeile hinzu. Zum Beispiel:

```
echo "## Running $className..."
```

Sobald Sie die Workflow-Ausführung starten, sollten Sie Ausgaben von beiden Workern und von der Workflow-Ausführung selbst sehen. Wenn der Workflow schließlich abgeschlossen ist, wird die Ausgabe auf dem Bildschirm angezeigt.

## Vollständiger Quellcode für dieses Beispiel

Du kannst den [kompletten Quellcode](#) für dieses Beispiel auf Github im `aws-java-developer-guideRepository` durchsuchen.

## Weitere Informationen

- Die hier gezeigten Worker können zu verloren gegangenen Aufgaben führen, wenn sie beendet werden, während noch eine Workflow-Abfrage läuft. Unter [Korrektes Herunterfahren von Aktivitäts- und Workflow-Workern](#) erfahren Sie, wie sich Worker korrekt herunterfahren lassen.
- Weitere Informationen Amazon SWF finden Sie auf der [Amazon SWF](#) Startseite oder im [Amazon SWF Developer Guide](#).
- Sie können AWS Flow Framework for Java verwenden, um mithilfe von Anmerkungen komplexere Workflows in einem eleganten Java-Stil zu schreiben. Weitere Informationen finden Sie im [AWS Flow Framework for Java Developer Guide](#).

## Lambda Aufgaben

Als Alternative zu oder in Verbindung mit Amazon SWF Aktivitäten können Sie [Lambda-Funktionen](#) verwenden, um Arbeitseinheiten in Ihren Workflows darzustellen und sie ähnlich wie Aktivitäten zu planen.

Dieses Thema konzentriert sich auf die Implementierung von Amazon SWF Lambda Aufgaben mithilfe von. AWS SDK für Java Weitere Informationen zu Lambda Aufgaben im Allgemeinen finden Sie unter [AWS Lambda Aufgaben](#) im Amazon SWF Entwicklerhandbuch.

## Einrichten einer serviceübergreifenden IAM-Rolle zum Ausführen Ihrer Lambda-Funktion

Bevor Sie Ihre Lambda Funktion ausführen Amazon SWF können, müssen Sie eine IAM-Rolle einrichten, um die Amazon SWF Erlaubnis zu erteilen, Lambda Funktionen in Ihrem Namen auszuführen. Vollständige Informationen dazu finden Sie unter [AWS Lambda Aufgaben](#).

Sie benötigen den Amazon-Ressourcennamen (ARN) dieser IAM-Rolle, wenn Sie einen Workflow registrieren, der Lambda Aufgaben verwendet.

### Erstellen Sie eine Funktion Lambda

Sie können Lambda Funktionen in einer Reihe verschiedener Sprachen schreiben, einschließlich Java. Vollständige Informationen zum Erstellen, Bereitstellen und Verwenden von Lambda Funktionen finden Sie im [AWS Lambda Entwicklerhandbuch](#).

#### Note

Es spielt keine Rolle, in welcher Sprache Sie Ihre Lambda Funktion schreiben, sie kann von jedem Amazon SWF Workflow geplant und ausgeführt werden, unabhängig von der Sprache, in der Ihr Workflow-Code geschrieben ist. Amazon SWF kümmert sich um die Einzelheiten der Ausführung der Funktion und der Weitergabe von Daten an und von ihr.

Hier ist eine einfache Lambda Funktion, die anstelle der Aktivität unter [Erstellen einer einfachen Amazon SWF Anwendung](#) verwendet werden könnte.

- Diese Version ist geschrieben JavaScript und kann direkt mit dem folgenden Befehl eingegeben werden [AWS-Managementkonsole](#):

```
exports.handler = function(event, context) {  
    context.succeed("Hello, " + event.who + "!");  
};
```

- Hier ist dieselbe in Java geschriebene Funktion, die Sie auch auf Lambda bereitstellen und ausführen könnten:

```
package example.swf.hello1lambda;
```

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.util.json.JSONException;
import com.amazonaws.util.json.JSONObject;

public class SwfHelloLambdaFunction implements RequestHandler<Object, Object> {
    @Override
    public Object handleRequest(Object input, Context context) {
        String who = "{SWF}";
        if (input != null) {
            JSONObject jso = null;
            try {
                jso = new JSONObject(input.toString());
                who = jso.getString("who");
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
        return ("Hello, " + who + "!");
    }
}
```

### Note

Weitere Informationen zur Bereitstellung von Java-Funktionen in Lambda finden Sie unter [Creating a Deployment Package \(Java\)](#) im AWS Lambda Developer Guide. Sie sollten sich auch den Abschnitt mit dem Titel [Programmiermodell für Lambda Autorenfunktionen in Java](#) ansehen.

Lambda Funktionen verwenden ein Ereignis oder Eingabeobjekt als ersten Parameter und ein Kontextobjekt als zweiten, das Informationen über die Anforderung zur Ausführung der Lambda Funktion bereitstellt. Diese bestimmte Funktion erwartet die Parameter im JSON-Format, wobei das Feld `who` auf den Namen zum Zusammensetzen der Begrüßung gesetzt ist.

## Registrieren Sie einen Workflow zur Verwendung mit Lambda

Damit ein Workflow eine Lambda Funktion planen kann, müssen Sie den Namen der IAM-Rolle angeben, die die Berechtigung zum Aufrufen Lambda von Funktionen erteilt `AmazonSWF`. Sie können dies bei der Workflow-Registrierung festlegen, indem Sie die `setDefaultLambdaRole` Methoden `withDefaultLambdaRole` oder `useDefaultLambdaRole` verwenden. [RegisterWorkflowTypeRequest](#)

```

System.out.println("*** Registering the workflow type '" + WORKFLOW + "-" +
    WORKFLOW_VERSION
    + "'.");
try {
    swf.registerWorkflowType(new RegisterWorkflowTypeRequest()
        .withDomain(DOMAIN)
        .withName(WORKFLOW)
        .withDefaultLambdaRole(lambda_role_arn)
        .withVersion(WORKFLOW_VERSION)
        .withDefaultChildPolicy(ChildPolicy.TERMINATE)
        .withDefaultTaskList(new TaskList().withName(TASKLIST))
        .withDefaultTaskStartToCloseTimeout("30"));
}
catch (TypeAlreadyExistsException e) {

```

## Planen Sie eine Lambda Aufgabe

Das Planen einer Lambda Aufgabe ähnelt dem Planen einer Aktivität. Sie geben eine [Entscheidung](#) mit einem `ScheduleLambdaFunction` `DecisionType` und mit an [ScheduleLambdaFunctionDecisionAttributes](#).

```

running_functions == 0 && scheduled_functions == 0) {
AWSLambda lam = AWSLambdaClientBuilder.defaultClient();
GetFunctionConfigurationResult function_config =
    lam.getFunctionConfiguration(
        new GetFunctionConfigurationRequest()
            .withFunctionName("HelloFunction"));
String function_arn = function_config.getFunctionArn();

ScheduleLambdaFunctionDecisionAttributes attrs =
    new ScheduleLambdaFunctionDecisionAttributes()
        .withId("HelloFunction (Lambda task example)")
        .withName(function_arn)
        .withInput(workflow_input);

decisions.add(

```

In der `ScheduleLambdaFunctionDecisionAttributes` müssen Sie einen Namen angeben, der der ARN der aufzurufenden Lambda Funktion ist, und eine ID, mit der die Lambda Funktion in Verlaufsprotokollen identifiziert Amazon SWF wird.

Sie können auch optionale Eingaben für die Lambda Funktion angeben und ihren Timeout-Wert für Start bis Ende festlegen. Dabei handelt es sich um die Anzahl der Sekunden, die die Lambda Funktion ausführen darf, bevor ein `LambdaFunctionTimedOut` Ereignis generiert wird.

### Note

Dieser Code verwendet den [AWSLambdaClient](#), um den ARN der Lambda Funktion anhand des Funktionsnamens abzurufen. Sie können diese Technik verwenden, um zu vermeiden, dass der vollständige ARN (einschließlich Ihrer AWS-Konto ID) in Ihrem Code fest codiert wird.

## Behandeln Sie Lambda-Funktionsereignisse in Ihrem Decider

Lambda Aufgaben generieren eine Reihe von Ereignissen, auf die Sie reagieren können, wenn Sie in Ihrem Workflow-Worker nach Entscheidungsaufgaben suchen, die dem Lebenszyklus Ihrer Lambda Aufgabe entsprechen, mit [EventType](#) Werten wie `LambdaFunctionScheduled`, `LambdaFunctionStarted` und `LambdaFunctionCompleted`. Wenn die Lambda Funktion fehlschlägt oder ihre Ausführung länger dauert als der eingestellte Timeout-Wert, erhalten Sie entweder einen Ereignistyp `LambdaFunctionFailed` oder einen `LambdaFunctionTimedOut` Ereignistyp.

```
boolean function_completed = false;
String result = null;

System.out.println("Executing the decision task for the history events: [");
for (HistoryEvent event : events) {
    System.out.println("  " + event);
    EventType event_type = EventType.fromValue(event.getEventType());
    switch(event_type) {
    case WorkflowExecutionStarted:
        workflow_input =
            event.getWorkflowExecutionStartedEventAttributes()
                .getInput();
        break;
    case LambdaFunctionScheduled:
        scheduled_functions++;
        break;
    case ScheduleLambdaFunctionFailed:
        scheduled_functions--;
    }
```

```
        break;
    case LambdaFunctionStarted:
        scheduled_functions--;
        running_functions++;
        break;
    case LambdaFunctionCompleted:
        running_functions--;
        function_completed = true;
        result = event.getLambdaFunctionCompletedEventAttributes()
            .getResult();
        break;
    case LambdaFunctionFailed:
        running_functions--;
        break;
    case LambdaFunctionTimedOut:
        running_functions--;
        break;
```

## Empfangen Sie die Ausgabe Ihrer Funktion Lambda

Wenn Sie einen Befehl `LambdaFunctionCompleted`` [EventType](#), you can retrieve your `0` function's return value by first calling ``getLambdaFunctionCompletedEventAttributes`` [HistoryEvent](#) zum Abrufen eines [LambdaFunctionCompletedEventAttributes](#) Objekts erhalten und dann dessen `getResult` Methode aufrufen, um die Ausgabe der Lambda Funktion abzurufen:

```
LambdaFunctionCompleted:
running_functions--;
```

## Vollständiger Quellcode für dieses Beispiel

Sie können die komplette Quellcode: `github: < awsdocs/aws-java-developer-guide/tree/master/doc_source/snippets/helloswf_lambda/>` nach diesem Beispiel auf Github im Repository durchsuchen. `aws-java-developer-guide`

## Korrektes Herunterfahren von Aktivitäts- und Workflow-Workern

Das Thema [Erstellen einer einfachen Amazon SWF Anwendung](#) bot eine vollständige Implementierung einer einfachen Workflow-Anwendung, die aus einer Registrierungsanwendung, einem Aktivitäts- und Workflow-Worker sowie einem Workflow-Starter bestand.

Worker-Klassen sind so konzipiert, dass sie kontinuierlich ausgeführt werden und nach Aufgaben suchen, die von Amazon SWF gesendet wurden, um Aktivitäten auszuführen oder Entscheidungen zurückzugeben. Sobald eine Umfrage angefordert wurde, wird der Abfragende Amazon SWF aufgezeichnet und versucht, ihm eine Aufgabe zuzuweisen.

Wenn der Workflow-Worker während einer langen Umfrage beendet wird, versucht er Amazon SWF möglicherweise trotzdem, eine Aufgabe an den abgebrochenen Mitarbeiter zu senden, was dazu führt, dass die Aufgabe verloren geht (bis die Aufgabe das Timeout erreicht).

Eine Möglichkeit zur Bewältigung dieser Situation besteht darin, zu warten, bis alle Long-Poll-Anforderungen zurückkehren, bevor der Worker beendet wird.

In diesem Thema schreiben wir den Aktivitäts-Worker von `helloworld` um und verwenden Java-Hooks für das Herunterfahren. So versuchen wir, den Aktivitäts-Worker ordnungsgemäß herunterzufahren.

Hier finden Sie den vollständigen Code:

```
import java.util.concurrent.CountDownLatch;
import java.util.concurrent.TimeUnit;

import com.amazonaws.regions.Regions;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.ActivityTask;
import com.amazonaws.services.simpleworkflow.model.PollForActivityTaskRequest;
import com.amazonaws.services.simpleworkflow.model.RespondActivityTaskCompletedRequest;
import com.amazonaws.services.simpleworkflow.model.RespondActivityTaskFailedRequest;
import com.amazonaws.services.simpleworkflow.model.TaskList;

public class ActivityWorkerWithGracefulShutdown {

    private static final AmazonSimpleWorkflow swf =

AmazonSimpleWorkflowClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
    private static final CountDownLatch waitForTermination = new CountDownLatch(1);
    private static volatile boolean terminate = false;

    private static String executeActivityTask(String input) throws Throwable {
        return "Hello, " + input + "!";
    }

    public static void main(String[] args) {
        Runtime.getRuntime().addShutdownHook(new Thread() {
```

```
@Override
public void run() {
    try {
        terminate = true;
        System.out.println("Waiting for the current poll request" +
            " to return before shutting down.");
        waitForTermination.await(60, TimeUnit.SECONDS);
    }
    catch (InterruptedException e) {
        // ignore
    }
}

});
try {
    pollAndExecute();
}
finally {
    waitForTermination.countDown();
}
}

public static void pollAndExecute() {
    while (!terminate) {
        System.out.println("Polling for an activity task from the tasklist '"
            + HelloTypes.TASKLIST + "' in the domain '"
            + HelloTypes.DOMAIN + "'.");

        ActivityTask task = swf.pollForActivityTask(new
PollForActivityTaskRequest()
            .withDomain(HelloTypes.DOMAIN)
            .withTaskList(new TaskList().withName(HelloTypes.TASKLIST)));

        String taskToken = task.getTaskToken();

        if (taskToken != null) {
            String result = null;
            Throwable error = null;

            try {
                System.out.println("Executing the activity task with input '"
                    + task.getInput() + "'.");
                result = executeActivityTask(task.getInput());
            }
            catch (Throwable th) {
```

```
        error = th;
    }

    if (error == null) {
        System.out.println("The activity task succeeded with result '"
            + result + "'.");
        swf.respondActivityTaskCompleted(
            new RespondActivityTaskCompletedRequest()
                .withTaskToken(taskToken)
                .withResult(result));
    }
    else {
        System.out.println("The activity task failed with the error '"
            + error.getClass().getSimpleName() + "'.");
        swf.respondActivityTaskFailed(
            new RespondActivityTaskFailedRequest()
                .withTaskToken(taskToken)
                .withReason(error.getClass().getSimpleName())
                .withDetails(error.getMessage()));
    }
}
}
}
```

In dieser Version wurde der Polling-Code aus der main-Funktion in der ursprünglichen Version in eine eigene Methode `pollAndExecute` verschoben.

Die main Funktion verwendet nun einen [CountDownLatch](#) in Verbindung mit einem [Shutdown-Hook](#), damit der Thread bis zu 60 Sekunden wartet, nachdem seine Beendigung angefordert wurde, bevor der Thread beendet wird.

## Registrieren von Domänen

Jeder Workflow und jede Aktivität in [Amazon SWF](#) benötigt eine Domain, in der sie ausgeführt werden kann.

1. Erstellen Sie ein neues [RegisterDomainRequest](#) Objekt und geben Sie ihm mindestens den Domännennamen und die Aufbewahrungsfrist für die Workflow-Ausführung an (diese Parameter sind beide erforderlich).
2. Rufen Sie die [AmazonSimpleWorkflowClient.registerDomain-Methode](#) mit dem Objekt auf `RegisterDomainRequest`.

3. Ermitteln Sie [DomainAlreadyExistsException](#), ob die Domain, die Sie anfordern, bereits existiert (in diesem Fall ist normalerweise keine Aktion erforderlich).

Der folgende Code veranschaulicht dieses Vorgehen:

```
public void register_swf_domain(AmazonSimpleWorkflowClient swf, String name)
{
    RegisterDomainRequest request = new RegisterDomainRequest().withName(name);
    request.setWorkflowExecutionRetentionPeriodInDays("10");
    try
    {
        swf.registerDomain(request);
    }
    catch (DomainAlreadyExistsException e)
    {
        System.out.println("Domain already exists!");
    }
}
```

## Auflisten von Domänen

Sie können die mit Ihrem Konto und Ihrer AWS Region verknüpften [Amazon SWF](#) Domains nach Registrierungstyp auflisten.

1. Erstellen Sie ein [ListDomainsRequest](#) Objekt und geben Sie den Registrierungsstatus der Domains an, an denen Sie interessiert sind — dies ist erforderlich.
2. Rufen Sie [AmazonSimpleWorkflowClient.listDomains](#) mit dem Objekt auf. ListDomainRequest Die Ergebnisse werden in einem [DomainInfos](#) Objekt bereitgestellt.
3. Rufen Sie das zurückgegebene Objekt [getDomainInfos](#) auf, um eine Liste von [DomainInfo](#) Objekten zu erhalten.
4. Rufen Sie [GetName](#) für jedes DomainInfo Objekt auf, um seinen Namen zu erhalten.

Der folgende Code veranschaulicht dieses Vorgehen:

```
public void list_swf_domains(AmazonSimpleWorkflowClient swf)
{
    ListDomainsRequest request = new ListDomainsRequest();
    request.setRegistrationStatus("REGISTERED");
    DomainInfos domains = swf.listDomains(request);
}
```

```
System.out.println("Current Domains:");
for (DomainInfo di : domains.getDomainInfos())
{
    System.out.println(" * " + di.getName());
}
}
```

## Im SDK enthaltene Codebeispiele

Das AWS SDK für Java wird mit Codebeispielen geliefert, die viele Funktionen des SDK in baubaren, ausführbaren Programmen demonstrieren. Sie können diese studieren oder ändern, um Ihre eigenen AWS Lösungen mithilfe von zu implementieren. AWS SDK für Java

### Abrufen der Beispiele

Die AWS SDK für Java Codebeispiele befinden sich im Beispielverzeichnis des SDK. Wenn Sie das SDK anhand der Informationen unter [Einrichten](#) von heruntergeladen und installiert haben AWS SDK für Java, haben Sie die Beispiele bereits auf Ihrem System.

Sie können sich die neuesten Beispiele auch im AWS SDK für Java GitHub Repository im Verzeichnis [src/samples](#) ansehen.

### Erstellen und Ausführen der Beispiele in der Befehlszeile

Mithilfe der enthaltenen [Ant](#)-Build-Skripts können Sie die Beispiele leicht erstellen und in der Befehlszeile ausführen. Jedes Beispiel enthält eine README-Datei im HTML-Format mit speziellen Informationen für das jeweilige Beispiel.

#### Note

Wenn Sie sich den Beispielcode ansehen GitHub, klicken Sie in der Quellcode-Anzeige auf die Schaltfläche Raw, wenn Sie sich die Datei README.html des Beispiels ansehen. Im Raw-Modus wird das HTML korrekt im Browser dargestellt.

### Voraussetzungen

Bevor Sie eines der AWS SDK für Java Beispiele ausführen, müssen Sie Ihre AWS Anmeldeinformationen in der Umgebung oder mit den einrichten AWS CLI, wie unter [AWS](#)

[Anmeldeinformationen einrichten und Region für die Entwicklung](#) angegeben, festlegen. Die Beispiele verwenden die standardmäßige Anbieterkette von Anmeldeinformationen, sofern möglich. Wenn Sie Ihre Anmeldeinformationen auf diese Weise festlegen, können Sie die riskante Praxis vermeiden, Ihre AWS Anmeldeinformationen in Dateien im Quellcodeverzeichnis einzufügen (wo sie versehentlich eingecheckt und öffentlich zugänglich gemacht werden könnten).

## Ausführen der Beispiele

1. Wechseln Sie in das Verzeichnis mit dem Beispiel-Code. Wenn Sie sich beispielsweise im Stammverzeichnis des AWS SDK-Downloads befinden und das `AwsConsoleApp` Beispiel ausführen möchten, geben Sie Folgendes ein:

```
cd samples/AwsConsoleApp
```

2. Erstellen und führen Sie das Beispiel mit Ant aus. Das Standard-Build-Ziel führt beide Aktionen aus, daher können Sie einfach Folgendes eingeben:

```
ant
```

Das Beispiel druckt Informationen auf die Standardausgabe, zum Beispiel:

```
=====
Welcome to the {AWS} Java SDK!
=====
You have access to 4 Availability Zones.

You have 0 {EC2} instance(s) running.

You have 13 Amazon SimpleDB domain(s) containing a total of 62 items.

You have 23 {S3} bucket(s), containing 44 objects with a total size of 154767691 bytes.
```

## Erstellen und Ausführen der Beispiele in der Eclipse-IDE

Wenn Sie das verwenden AWS Toolkit for Eclipse, können Sie auch ein neues Projekt in Eclipse starten, das auf dem basiert, AWS SDK für Java oder das SDK zu einem vorhandenen Java-Projekt hinzufügen.

## Voraussetzungen

Nach der Installation von empfehlen wir AWS Toolkit for Eclipse, das Toolkit mit Ihren Sicherheitsanmeldedaten zu konfigurieren. Sie können dies jederzeit tun, indem Sie in Eclipse im Menü „Fenster“ die Option „Einstellungen“ und dann den Abschnitt „AWS Toolkit“ auswählen.

## Ausführen der Beispiele

1. Öffnen Sie Eclipse.
2. Erstellen Sie ein neues AWS Java-Projekt. Klicken Sie in Eclipse im Menü Datei auf Neu und dann auf Projekt. Der Assistent Neues Projekt wird geöffnet.
3. Erweitern Sie die AWS Kategorie und wählen Sie dann AWS Java-Projekt aus.
4. Wählen Sie Weiter. Die Seite "Projekteinstellungen" wird angezeigt.
5. Geben Sie einen Namen in das Feld Projektname ein. In der Gruppe AWS SDK für Java Beispiele werden die im SDK verfügbaren Beispiele angezeigt, wie zuvor beschrieben.
6. Wählen Sie durch Markieren der Kontrollkästchen die Beispiele aus, die Sie in Ihr Projekt aufnehmen möchten.
7. Geben Sie Ihre AWS Anmeldedaten ein. Wenn Sie das bereits AWS Toolkit for Eclipse mit Ihren Anmeldeinformationen konfiguriert haben, wird dies automatisch ausgefüllt.
8. Wählen Sie Finish (Abschließen). Das Projekt wird erstellt und zum Projekt-Explorer hinzugefügt.
9. Wählen Sie die `.java`-Beispieldatei aus, die Sie ausführen möchten. Wählen Sie für das Amazon S3 Beispiel beispielsweise `S3Sample.java`.
10. Klicken Sie im Menü Ausführen auf Ausführen.
11. Klicken Sie im Projekt-Explorer mit der rechten Maustaste auf das Projekt, zeigen Sie dann auf Build-Pfad und wählen Sie Bibliotheken hinzufügen.
12. Wählen Sie AWS Java SDK, wählen Sie Weiter und folgen Sie dann den weiteren Anweisungen auf dem Bildschirm.

# Sicherheit für die AWS SDK für 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 gemeinsame Verantwortung zwischen AWS Ihnen und Ihnen. Im [Modell der übergreifenden Verantwortlichkeit](#) wird Folgendes mit „Sicherheit der Cloud“ bzw. „Sicherheit in der Cloud“ umschrieben:

**Sicherheit der Cloud** — AWS ist verantwortlich für den Schutz der Infrastruktur, auf der alle in der AWS Cloud angebotenen Dienste ausgeführt werden, und für die Bereitstellung von Diensten, die Sie sicher nutzen können. Unsere Sicherheitsverantwortung hat bei uns höchste Priorität AWS, und die Wirksamkeit unserer Sicherheit wird im Rahmen der [AWS Compliance-Programme](#) regelmäßig von externen Prüfern getestet und verifiziert.

**Sicherheit in der Cloud** — Ihre Verantwortung richtet sich nach dem von Ihnen genutzten AWS Dienst und anderen Faktoren, wie der Sensibilität Ihrer Daten, den Anforderungen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften.

Dieses AWS Produkt oder dieser Service folgt dem [Modell der gemeinsamen Verantwortung](#) in Bezug auf die spezifischen Amazon Web Services (AWS) -Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der [Seite mit der Dokumentation zur AWS Servicesicherheit](#) und den [AWS Services, für die das AWS Compliance-Programm zur Einhaltung der](#) Vorschriften zuständig ist.

## Themen

- [Datenschutz in AWS SDK für Java 1.x](#)
- [AWS SDK für Java Unterstützung für TLS](#)
- [Identitäts- und Zugriffsverwaltung](#)
- [Konformitätsvalidierung dafür AWS Produkt oder Dienstleistung](#)
- [Resilienz dafür AWS Produkt oder Dienstleistung](#)
- [Sicherheit der Infrastruktur in diesem Bereich AWS Produkt oder Service](#)
- [Amazon S3 Migration des Verschlüsselungsclients](#)

# Datenschutz in AWS SDK für Java 1.x

Das [Modell der gemeinsamen Verantwortung](#) gilt für den Datenschutz in diesem AWS Produkt oder dieser Dienstleistung. Wie in diesem Modell beschrieben, AWS ist es für den Schutz der globalen Infrastruktur verantwortlich, auf der die gesamte AWS Cloud läuft. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Dieser Inhalt enthält die Sicherheitskonfigurations- und Verwaltungsaufgaben für die von Ihnen verwendeten AWS -Services. Weitere Informationen zum Datenschutz finden Sie unter [Häufig gestellte Fragen zum Datenschutz](#). Informationen zum Datenschutz in Europa finden Sie im Blogbeitrag [AWS Shared Responsibility Model und GDPR](#) auf dem AWS Security Blog.

Aus Datenschutzgründen empfehlen wir Ihnen, Ihre AWS-Konto Anmeldeinformationen zu schützen und individuelle Benutzerkonten mit AWS Identity and Access Management (IAM) einzurichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem sollten Sie die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail.
- Verwenden Sie AWS Verschlüsselungslösungen mit allen Standardsicherheitskontrollen innerhalb der AWS Dienste.
- Verwenden Sie fortschrittliche verwaltete Sicherheitsdienste wie Amazon Macie, die Sie bei der Erkennung und Sicherung personenbezogener Daten unterstützen, die in Amazon S3 gespeichert sind.
- Wenn Sie für den Zugriff 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 wie z. B. im Feld Name keine sensiblen, identifizierenden Informationen wie Kontonummern von Kunden einzugeben. Dies gilt auch, wenn Sie mit diesem AWS Produkt oder Service oder anderen AWS Diensten über die Konsole, API oder SDKs arbeiten. AWS CLI AWS Alle Daten, die Sie in dieses AWS Produkt, diesen Service oder andere Dienste eingeben, werden möglicherweise zur Aufnahme in Diagnoseprotokolle aufgenommen. Wenn Sie eine URL für einen externen Server bereitstellen, schließen Sie keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL ein.

# AWS SDK für Java Unterstützung für TLS

Die folgenden Informationen gelten nur für die Java-SSL-Implementierung (die Standard-SSL-Implementierung in der AWS SDK für Java). Wenn Sie eine andere SSL-Implementierung verwenden, erfahren Sie in Ihrer spezifischen SSL-Implementierung, wie Sie TLS-Versionen erzwingen.

## Vorgehensweise zum Überprüfen der TLS-Version

Schlagen Sie in der Dokumentation Ihres JVM-Anbieters (Java Virtual Machine) nach, welche TLS-Versionen auf Ihrer Plattform unterstützt werden. Bei einigen JVMs gibt der folgende Code aus, welche SSL-Versionen unterstützt werden.

```
System.out.println(Arrays.toString(SSLContext.getDefault().getSupportedSSLParameters().getProtocols()));
```

Um den SSL-Handshake in Aktion zu sehen und welche Version von TLS verwendet wird, können Sie die Systemeigenschaft `javax.net.debug` verwenden.

```
java app.jar -Djavax.net.debug=ssl
```

### Note

TLS 1.3 ist nicht kompatibel mit SDK for Java Java-Versionen 1.9.5 bis 1.10.31. Weitere Informationen finden Sie im folgenden Blogbeitrag.

<https://aws.amazon.com/blogs/developer/tls-1-3-incompatibility-with-aws-sdk-for-java-versions-1-9-5-to-1-10-31/>

## Erzwingen einer Mindest-TLS-Version

Das SDK bevorzugt immer die neueste TLS-Version, die von der Plattform und dem Dienst unterstützt wird. Wenn Sie eine bestimmte TLS-Mindestversion erzwingen möchten, lesen Sie in der Dokumentation Ihrer JVM nach. Für OpenJDK-based JVMs können Sie die Systemeigenschaft verwenden. `jdk.tls.client.protocols`

```
java app.jar -Djdk.tls.client.protocols=PROTOCOLS
```

Die unterstützten Werte von PROTOCOLS finden Sie in der Dokumentation Ihrer JVM.

## Identitäts- und Zugriffsverwaltung

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service, den Zugriff auf Ressourcen sicher zu AWS kontrollieren. IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um Ressourcen zu verwenden. AWS IAM ist ein Programm AWS-Service, das Sie ohne zusätzliche Kosten nutzen können.

Themen

- [Zielgruppe](#)
- [Authentifizierung mit Identitäten](#)
- [Verwalten des Zugriffs mit Richtlinien](#)
- [Wie AWS-Services arbeiten Sie mit IAM](#)
- [Fehlerbehebung AWS Identität und Zugriff](#)

### Zielgruppe

Die Art und Weise, wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von der Arbeit ab, in der Sie tätig sind. AWS

**Dienstbenutzer** — Wenn Sie dies AWS-Services für Ihre Arbeit verwenden, stellt Ihnen Ihr Administrator die erforderlichen Anmeldeinformationen und Berechtigungen zur Verfügung. Wenn Sie für Ihre Arbeit mehr AWS Funktionen verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Funktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anfordern müssen. Falls Sie auf eine Funktion nicht zugreifen können AWS, finden [Fehlerbehebung AWS Identität und Zugriff](#) Sie weitere Informationen in der Bedienungsanleitung der von AWS-Service Ihnen verwendeten.

**Serviceadministrator** — Wenn Sie in Ihrem Unternehmen für die AWS Ressourcen verantwortlich sind, haben Sie wahrscheinlich vollen Zugriff auf AWS. Es ist Ihre Aufgabe, zu bestimmen, auf welche AWS Funktionen und Ressourcen Ihre Servicebenutzer zugreifen sollen. Anschließend müssen Sie Anforderungen an Ihren IAM-Administrator senden, um die Berechtigungen der Servicebenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen darüber, wie Ihr Unternehmen IAM verwenden kann AWS, finden Sie in der Benutzeranleitung des von AWS-Service Ihnen verwendeten.

IAM-Administrator: Wenn Sie als IAM-Administrator fungieren, sollten Sie Einzelheiten dazu kennen, wie Sie Richtlinien zur Verwaltung des Zugriffs auf AWS verfassen können. Beispiele für AWS identitätsbasierte Richtlinien, die Sie in IAM verwenden können, finden Sie im Benutzerhandbuch der AWS-Service von Ihnen verwendeten.

## Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen sich als IAM-Benutzer authentifizieren oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich als föderierte Identität anmelden, indem Sie Anmeldeinformationen aus einer Identitätsquelle wie AWS IAM Identity Center (IAM Identity Center), Single Sign-On-Authentifizierung oder Anmeldeinformationen verwenden. Google/Facebook Weitere Informationen zum Anmelden finden Sie unter [So melden Sie sich bei Ihrem AWS-Konto an](#) im Benutzerhandbuch für AWS-Anmeldung .

AWS Bietet für den programmatischen Zugriff ein SDK und eine CLI zum kryptografischen Signieren von Anfragen. Weitere Informationen finden Sie unter [AWS Signature Version 4 for API requests](#) im IAM-Benutzerhandbuch.

### AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, dem sogenannten AWS-Konto Root-Benutzer, der vollständigen Zugriff auf alle AWS-Services Ressourcen hat. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Eine Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter [Tasks that require root user credentials](#) im IAM-Benutzerhandbuch.

### Verbundidentität

Es hat sich bewährt, dass menschliche Benutzer für den Zugriff AWS-Services mithilfe temporärer Anmeldeinformationen einen Verbund mit einem Identitätsanbieter verwenden müssen.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensverzeichnis, Ihrem Directory Service Web-Identitätsanbieter oder der AWS-Services mithilfe von Anmeldeinformationen aus einer Identitätsquelle zugreift. Verbundene Identitäten übernehmen Rollen, die temporäre Anmeldeinformationen bereitstellen.

Für die zentrale Zugriffsverwaltung empfehlen wir AWS IAM Identity Center. Weitere Informationen finden Sie unter [Was ist IAM Identity Center?](#) im AWS IAM Identity Center -Benutzerhandbuch.

## IAM-Benutzer und -Gruppen

Ein [IAM-Benutzer](#) ist eine Identität mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Wir empfehlen die Verwendung temporärer Anmeldeinformationen anstelle von IAM-Benutzern mit langfristigen Anmeldeinformationen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter [Erfordern, dass menschliche Benutzer den Verbund mit einem Identitätsanbieter verwenden müssen, um AWS mithilfe temporärer Anmeldeinformationen darauf zugreifen zu können](#).

Eine [IAM-Gruppe](#) spezifiziert eine Sammlung von IAM-Benutzern und erleichtert die Verwaltung von Berechtigungen für große Gruppen von Benutzern. Weitere Informationen finden Sie unter [Anwendungsfälle für IAM-Benutzer](#) im IAM-Benutzerhandbuch.

## IAM-Rollen

Eine [IAM-Rolle](#) ist eine Identität mit spezifischen Berechtigungen, die temporäre Anmeldeinformationen bereitstellt. Sie können eine Rolle übernehmen, indem Sie [von einer Benutzer- zu einer IAM-Rolle \(Konsole\) wechseln](#) AWS CLI oder einen AWS API-Vorgang aufrufen. Weitere Informationen finden Sie unter [Methoden, um eine Rolle zu übernehmen](#) im IAM-Benutzerhandbuch.

IAM-Rollen sind nützlich für den Verbundbenutzer-Zugriff, temporäre IAM-Benutzerberechtigungen, kontoübergreifenden Zugriff, serviceübergreifenden Zugriff und Anwendungen, die auf Amazon EC2 laufen. Weitere Informationen finden Sie unter [Kontoübergreifender Ressourcenzugriff in IAM](#) im IAM-Benutzerhandbuch.

## Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie definiert Berechtigungen, wenn sie mit einer Identität oder Ressource verknüpft sind. AWS bewertet diese Richtlinien, wenn ein Principal eine Anfrage stellt. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu JSON-Richtliniendokumenten finden Sie unter [Übersicht über JSON-Richtlinien](#) im IAM-Benutzerhandbuch.

Mit Hilfe von Richtlinien legen Administratoren fest, wer Zugriff auf was hat, indem sie definieren, welches Prinzipal welche Aktionen auf welchen Ressourcen und unter welchen Bedingungendurchführen darf.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator erstellt IAM-Richtlinien und fügt sie zu Rollen hinzu, die die Benutzer dann übernehmen können. IAM-Richtlinien definieren Berechtigungen unabhängig von der Methode, die zur Ausführung der Operation verwendet wird.

## Identity-based Richtlinien

Identity-based Richtlinien sind Richtliniendokumente für JSON-Berechtigungen, die Sie an eine Identität (Benutzer, Gruppe oder Rolle) anhängen. Diese Richtlinien steuern, welche Aktionen Identitäten für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter [Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien](#) im IAM-Benutzerhandbuch.

Identity-based Richtlinien können Inline-Richtlinien (direkt in eine einzelne Identität eingebettet) oder verwaltete Richtlinien (eigenständige Richtlinien, die mehreren Identitäten zugeordnet sind) sein. Informationen dazu, wie Sie zwischen verwalteten und Inline-Richtlinien wählen, finden Sie unter [Choose between managed policies and inline policies](#) im IAM-Benutzerhandbuch.

## Resource-based Richtlinien

Resource-based Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anhängen. Beispiele hierfür sind Vertrauensrichtlinien für IAM-Rollen und Amazon S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Sie müssen in einer ressourcenbasierten Richtlinie [einen Prinzipal angeben](#).

Resource-based Richtlinien sind Inline-Richtlinien, die sich in diesem Dienst befinden. 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, die ACLs unterstützen. AWS WAF Weitere Informationen“ zu ACLs finden Sie unter [Zugriffskontrollliste \(ACL\) – Übersicht](#) (Access Control List) im Amazon-Simple-Storage-Service-Entwicklerhandbuch.

## Weitere Richtlinientypen

AWS unterstützt zusätzliche Richtlinientypen, mit denen die maximalen Berechtigungen festgelegt werden können, die durch gängigere Richtlinientypen gewährt werden:

- **Berechtigungsgrenzen** – Eine Berechtigungsgrenze legt die maximalen Berechtigungen fest, die eine identitätsbasierte Richtlinie einer IAM-Entität erteilen kann. Weitere Informationen finden Sie unter [Berechtigungsgrenzen für IAM-Entitäten](#) im -IAM-Benutzerhandbuch.
- **Service-Kontrollrichtlinien (SCPs)** – SCPs legen die maximalen Berechtigungen für eine Organisation oder Organisationseinheit in AWS Organizations fest. Weitere Informationen finden Sie unter [Service-Kontrollrichtlinien](#) im AWS Organizations -Benutzerhandbuch.
- **Ressourcen-Kontrollrichtlinien (RCPs)** – RCPs definieren die maximale Anzahl an Berechtigungen, die Ressourcen in Ihren Konten zur Verfügung stehen. Weitere Informationen finden Sie unter [Ressourcen-Kontrollrichtlinien](#) im AWS Organizations -Benutzerhandbuch.
- **Sitzungsrichtlinien** – Sitzungsrichtlinien sind erweiterte Richtlinien, die als Parameter übergeben werden, wenn Sie eine temporäre Sitzung für eine Rolle oder einen Verbundbenutzer erstellen. Weitere Informationen finden Sie unter [Sitzungsrichtlinien](#) im IAM-Benutzerhandbuch.

## Mehrere Richtlinientypen

Wenn mehrere Arten von Richtlinien für eine Anfrage gelten, sind die daraus resultierenden Berechtigungen schwieriger zu verstehen. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie unter [Bewertungslogik für Richtlinien](#) im IAM-Benutzerhandbuch.

## Wie AWS-Services arbeiten Sie mit IAM

Einen allgemeinen Überblick darüber, wie die meisten IAM-Funktionen AWS-Services funktionieren, finden Sie im [AWS IAM-Benutzerhandbuch unter Dienste, die mit IAM funktionieren](#).

Informationen zur Verwendung bestimmter Dienste AWS-Service mit IAM finden Sie im Abschnitt Sicherheit im Benutzerhandbuch des jeweiligen Dienstes.

## Fehlerbehebung AWS Identität und Zugriff

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit AWS und IAM auftreten können.

## Themen

- [Ich bin nicht berechtigt, eine Aktion durchzuführen in AWS](#)
- [Ich bin nicht berechtigt, iam auszuführen: PassRole](#)
- [Ich möchte Personen außerhalb meiner Umgebung zulassen AWS-Konto um auf meine zuzugreifen AWS Ressourcen](#)

### Ich bin nicht berechtigt, eine Aktion durchzuführen in AWS

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht zur Durchführung einer Aktion berechtigt sind, müssen Ihre Richtlinien aktualisiert werden, damit Sie die Aktion durchführen können.

Der folgende Beispielfehler tritt auf, wenn der IAM-Benutzer `mateojackson` versucht, über die Konsole Details zu einer fiktiven `my-example-widget`-Ressource anzuzeigen, jedoch nicht über `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 berechtigt, iam auszuführen: PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zum Durchführen der `iam:PassRole`-Aktion autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um eine Rolle an AWS übergeben zu können.

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Dienst zu übergeben, anstatt eine neue Servicerolle oder eine dienstverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in AWS auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

## Ich möchte Personen außerhalb meiner Umgebung zulassen AWS-Konto um auf meine zuzugreifen AWS Ressourcen

Sie können eine Rolle erstellen, mit der Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation auf Ihre Ressourcen zugreifen können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Im Fall von Diensten, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (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 darüber, ob diese Funktionen AWS unterstützt werden, finden Sie unter [Wie AWS-Services arbeiten Sie mit IAM](#).
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im IAM-Benutzerhandbuch unter [Gewähren des Zugriffs auf einen IAM-Benutzer in einem anderen AWS-Konto, den Sie besitzen](#).
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie [AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte](#).
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter [Gewähren von Zugriff für extern authentifizierte Benutzer \(Identitätsverbund\)](#) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter [Kontoübergreifender Ressourcenzugriff in IAM](#) im IAM-Benutzerhandbuch.

## Konformitätsvalidierung dafür AWS Produkt oder Dienstleistung

Informationen darüber, ob AWS-Service ein [AWS-Services in den Geltungsbereich bestimmter Compliance-Programme fällt](#), finden Sie unter [Umfang nach Compliance-Programm AWS-Services unter](#) . Wählen Sie dort das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter [AWS Compliance-Programme AWS](#) .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter [Berichte herunterladen unter](#) .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. Weitere Informationen zu Ihrer Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services finden Sie in der [AWS Sicherheitsdokumentation](#).

Dieses AWS Produkt oder dieser Service folgt dem [Modell der gemeinsamen Verantwortung](#) in Bezug auf die spezifischen Amazon Web Services (AWS) -Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der [Seite mit der Dokumentation zur AWS Servicesicherheit](#) und den [AWS Services, für die das AWS Compliance-Programm zur Einhaltung der](#) Vorschriften zuständig ist.

## Resilienz dafür AWS Produkt oder Dienstleistung

Die AWS globale Infrastruktur basiert auf AWS-Regionen Availability Zones.

AWS-Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind.

Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter [AWS Globale Infrastruktur](#).

Dieses AWS Produkt oder dieser Service folgt dem [Modell der gemeinsamen Verantwortung](#) in Bezug auf die spezifischen Amazon Web Services (AWS) -Services, die es unterstützt. Informationen

zur AWS Servicesicherheit finden Sie auf der [Seite mit der Dokumentation zur AWS Servicesicherheit](#) und den [AWS Services, für die das AWS Compliance-Programm zur Einhaltung der](#) Vorschriften zuständig ist.

## Sicherheit der Infrastruktur in diesem Bereich AWS Produkt oder Service

Dieses AWS Produkt oder dieser Dienst verwendet Managed Services und ist daher durch die AWS globale Netzwerksicherheit geschützt. Informationen zu AWS Sicherheitsdiensten und zum AWS Schutz der Infrastruktur finden Sie unter [AWS Cloud-Sicherheit](#). Informationen zum Entwerfen Ihrer AWS Umgebung unter Verwendung der bewährten Methoden für die Infrastruktursicherheit finden Sie unter [Infrastructure Protection](#) in Security Pillar AWS Well-Architected Framework.

Sie verwenden AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf dieses AWS Produkt oder diesen Service zuzugreifen. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Cipher-Suites mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman Die meisten modernen Systemen wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit [AWS -Security-Token-Service](#) (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Dieses AWS Produkt oder dieser Service folgt dem [Modell der gemeinsamen Verantwortung](#) in Bezug auf die spezifischen Amazon Web Services (AWS) -Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der [Seite mit der Dokumentation zur AWS Servicesicherheit](#) und den [AWS Services, für die das AWS Compliance-Programm zur Einhaltung der](#) Vorschriften zuständig ist.

## Amazon S3 Migration des Verschlüsselungsclients

In diesem Thema erfahren Sie, wie Sie Ihre Anwendungen von Version 1 (V1) des Verschlüsselungsclients Amazon Simple Storage Service (Amazon S3) auf Version 2 (V2) migrieren und die Anwendungsverfügbarkeit während des gesamten Migrationsprozesses sicherstellen.

## Voraussetzungen

Amazon S3 Für die clientseitige Verschlüsselung ist Folgendes erforderlich:

- Java 8 oder höher ist in Ihrer Anwendungsumgebung installiert. [Das AWS SDK für Java funktioniert mit dem Oracle Java SE Development Kit und mit Distributionen von Open Java Development Kit \(OpenJDK\) wie Amazon CorrettoRed Hat OpenJDK und JDK. AdoptOpen](#)
- [Das Bouncy Castle Crypto-Paket](#). Sie können die Bouncy Castle-.jar-Datei im Klassenpfad Ihrer Anwendungsumgebung platzieren oder Ihrer Maven-Datei eine Abhängigkeit von der ArtifactID `bcprov-ext-jdk15on` (mit der GroupID von) `org.bouncycastle pom.xml` hinzufügen.

## Überblick über die Migration

Diese Migration erfolgt in zwei Phasen:

1. Aktualisieren Sie bestehende Clients, damit sie neue Formate lesen können. Aktualisieren Sie Ihre Anwendung so, dass sie Version 1.11.837 oder höher verwendet, AWS SDK für Java und stellen Sie die Anwendung erneut bereit. Dadurch können die Amazon S3 clientseitigen Verschlüsselungsdienstclients in Ihrer Anwendung Objekte entschlüsseln, die von V2-Dienstclients erstellt wurden. Wenn Ihre Anwendung mehrere AWS SDKs verwendet, müssen Sie jedes SDK separat aktualisieren.
2. Migrieren Sie Verschlüsselungs- und Entschlüsselungscients auf V2. Sobald alle Ihre V1-Verschlüsselungscients die V2-Verschlüsselungsformate lesen können, aktualisieren Sie die Amazon S3 clientseitigen Verschlüsselungs- und Entschlüsselungscients in Ihrem Anwendungscode, sodass sie ihre V2-Entsprechungen verwenden.

## Aktualisieren Sie bestehende Clients, sodass sie neue Formate lesen können

Der V2-Verschlüsselungscient verwendet Verschlüsselungsalgorithmen, die ältere Versionen von AWS SDK für Java nicht unterstützen.

Der erste Schritt der Migration besteht darin, Ihre V1-Verschlüsselungscients so zu aktualisieren, dass sie Version 1.11.837 oder höher von verwenden. AWS SDK für Java(Wir empfehlen Ihnen, auf die neueste Release-Version zu aktualisieren, die Sie in der [Java API-Referenz Version 1.x](#) finden.) Aktualisieren Sie dazu die Abhängigkeit in Ihrer Projektkonfiguration. Nachdem Ihre Projektkonfiguration aktualisiert wurde, erstellen Sie Ihr Projekt neu und stellen Sie es erneut bereit.

Sobald Sie diese Schritte abgeschlossen haben, können die V1-Verschlüsselungsclients Ihrer Anwendung Objekte lesen, die von V2-Verschlüsselungsclients geschrieben wurden.

## Aktualisieren Sie die Abhängigkeit in Ihrer Projektkonfiguration

Ändern Sie Ihre Projektkonfigurationsdatei (z. B. pom.xml oder build.gradle), um Version 1.11.837 oder höher von zu verwenden. AWS SDK für Java Erstellen Sie dann Ihr Projekt neu und stellen Sie es erneut bereit.

Wenn Sie diesen Schritt vor der Bereitstellung des neuen Anwendungscodes abschließen, können Sie sicherstellen, dass die Verschlüsselungs- und Entschlüsselungsvorgänge während des Migrationsprozesses in Ihrer gesamten Flotte konsistent bleiben.

### Beispiel für die Verwendung von Maven

Ausschnitt aus einer Datei pom.xml:

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-bom</artifactId>
      <version>1.11.837</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

### Beispiel mit Gradle

Ausschnitt aus einer build.gradle-Datei:

```
dependencies {
  implementation platform('com.amazonaws:aws-java-sdk-bom:1.11.837')
  implementation 'com.amazonaws:aws-java-sdk-s3'
}
```

## Migrieren Sie Verschlüsselungs- und Entschlüsselungsclients auf V2

Sobald Ihr Projekt mit der neuesten SDK-Version aktualisiert wurde, können Sie Ihren Anwendungscode ändern, um den V2-Client zu verwenden. Aktualisieren Sie dazu

zunächst Ihren Code, um den neuen Service Client Builder zu verwenden. Stellen Sie dann Verschlüsselungsmaterial mithilfe einer Methode auf dem Builder bereit, die umbenannt wurde, und konfigurieren Sie Ihren Service Client nach Bedarf weiter.

Diese Codefragmente veranschaulichen, wie die clientseitige Verschlüsselung mit dem verwendet wird AWS SDK für Java, und bieten Vergleiche zwischen den V1- und V2-Verschlüsselungsclients.

## V1

```
// minimal configuration in V1; default CryptoMode.EncryptionOnly.  
EncryptionMaterialsProvider encryptionMaterialsProvider = ...  
AmazonS3Encryption encryptionClient = AmazonS3EncryptionClient.encryptionBuilder()  
    .withEncryptionMaterials(encryptionMaterialsProvider)  
    .build();
```

## V2

```
// minimal configuration in V2; default CryptoMode.StrictAuthenticatedEncryption.  
EncryptionMaterialsProvider encryptionMaterialsProvider = ...  
AmazonS3EncryptionV2 encryptionClient = AmazonS3EncryptionClientV2.encryptionBuilder()  
    .withEncryptionMaterialsProvider(encryptionMaterialsProvider)  
    .withCryptoConfiguration(new CryptoConfigurationV2()  
        // The following setting allows the client to read V1  
        encrypted objects  
        .withCryptoMode(CryptoMode.AuthenticatedEncryption)  
    )  
    .build();
```

Das obige Beispiel setzt den Wert `cryptoMode` auf `AuthenticatedEncryption`. Dies ist eine Einstellung, die es einem V2-Verschlüsselungsclient ermöglicht, Objekte zu lesen, die von einem V1-Verschlüsselungsclient geschrieben wurden. Wenn Ihr Client nicht die Fähigkeit benötigt, Objekte zu lesen, die von einem V1-Client geschrieben wurden, empfehlen wir, `StrictAuthenticatedEncryption` stattdessen die Standardeinstellung von zu verwenden.

## Konstruieren Sie einen V2-Verschlüsselungsclient

Der V2-Verschlüsselungsclient kann durch Aufrufen von `AmazonS3EncryptionClientV2.encryptionBuilder()` erstellt werden.

Sie können alle Ihre vorhandenen V1-Verschlüsselungsclients durch V2-Verschlüsselungsclients ersetzen. Ein V2-Verschlüsselungsclient kann immer jedes Objekt lesen, das von

einem V1-Verschlüsselungsclient geschrieben wurde, solange Sie dies zulassen, indem Sie den V2-Verschlüsselungsclient für die Verwendung von `AuthenticatedEncryption`cryptoMode`.

Das Erstellen eines neuen V2-Verschlüsselungsclients ist dem Erstellen eines V1-Verschlüsselungsclients sehr ähnlich. Es gibt jedoch einige Unterschiede:

- Sie werden ein `CryptoConfigurationV2` Objekt anstelle eines `CryptoConfiguration` Objekts verwenden, um den Client zu konfigurieren. Dieser Parameter muss angegeben werden.
- Die `cryptoMode` Standardeinstellung für den V2-Verschlüsselungsclient ist `StrictAuthenticatedEncryption`. Für den V1-Verschlüsselungsclient ist dies der `FallEncryptionOnly`.
- Die Methode `withEncryptionMaterials()` im Encryption Client Builder wurde in `withEncryptionMaterialsProvider()` umbenannt. Dies ist lediglich eine kosmetische Änderung, die den Argumenttyp genauer wiedergibt. Sie müssen die neue Methode verwenden, wenn Sie Ihren Service-Client konfigurieren.

#### Note

Lesen Sie beim Entschlüsseln mit AES-GCM das gesamte Objekt bis zum Ende, bevor Sie die entschlüsselten Daten verwenden. Dadurch wird überprüft, ob das Objekt seit der Verschlüsselung nicht geändert wurde.

## Verwenden Sie Anbieter von Verschlüsselungsmaterialien

Sie können weiterhin dieselben Anbieter für Verschlüsselungsmaterialien und Objekte für Verschlüsselungsmaterialien verwenden, die Sie bereits mit dem V1-Verschlüsselungsclient verwenden. Diese Klassen sind für die Bereitstellung der Schlüssel verantwortlich, die der Verschlüsselungsclient zum Schutz Ihrer Daten verwendet. Sie können sowohl mit dem V2- als auch mit dem V1-Verschlüsselungsclient synonym verwendet werden.

## Konfigurieren Sie den V2 Encryption Client

Der V2-Verschlüsselungsclient ist mit einem `CryptoConfigurationV2` Objekt konfiguriert. Dieses Objekt kann erstellt werden, indem sein Standardkonstruktor aufgerufen und dann seine Eigenschaften entsprechend den Standardeinstellungen geändert werden.

Die Standardwerte für `CryptoConfigurationV2` sind:

- `cryptoMode = CryptoMode.StrictAuthenticatedEncryption`
- `storageMode = CryptoStorageMode.ObjectMetadata`
- `secureRandom =` Instanz von `SecureRandom`
- `rangeGetMode = CryptoRangeGetMode.DISABLED`
- `unsafeUndecryptableObjectPassthrough = false`

Beachten Sie, `EncryptionOnly` dass dies `cryptoMode` im V2-Verschlüsselungsclient nicht unterstützt wird. Der V2-Verschlüsselungsclient verschlüsselt Inhalte immer mit authentifizierter Verschlüsselung und schützt Inhaltsverschlüsselungsschlüssel (CEKs) mithilfe von V2-Objekten. `KeyWrap`

Das folgende Beispiel zeigt, wie die Kryptokonfiguration in V1 angegeben wird und wie ein `CryptoConfigurationV2`-Objekt instanziiert wird, um es an den V2-Verschlüsselungs-Client-Builder zu übergeben.

V1

```
CryptoConfiguration cryptoConfiguration = new CryptoConfiguration()
    .withCryptoMode(CryptoMode.StrictAuthenticatedEncryption);
```

V2

```
CryptoConfigurationV2 cryptoConfiguration = new CryptoConfigurationV2()
    .withCryptoMode(CryptoMode.StrictAuthenticatedEncryption);
```

## Weitere Beispiele

Die folgenden Beispiele zeigen, wie bestimmte Anwendungsfälle im Zusammenhang mit einer Migration von V1 zu V2 behandelt werden können.

Konfigurieren Sie einen Service Client zum Lesen von Objekten, die vom V1 Encryption Client erstellt wurden

Um Objekte zu lesen, die zuvor mit einem V1-Verschlüsselungsclient geschrieben wurden, setzen Sie den `cryptoMode` Wert auf `AuthenticatedEncryption`. Der folgende Codeausschnitt zeigt, wie ein Konfigurationsobjekt mit dieser Einstellung erstellt wird.

```
CryptoConfigurationV2 cryptoConfiguration = new CryptoConfigurationV2()  
    .withCryptoMode(CryptoMode.AuthenticatedEncryption);
```

## Konfigurieren Sie einen Service Client für das Abrufen von Bytebereichen von Objekten

Um aus get einem verschlüsselten S3-Objekt auf einen Bytebereich zugreifen zu können, aktivieren Sie die neue Konfigurationseinstellung `rangeGetMode`. Diese Einstellung ist auf dem V2-Verschlüsselungsclient standardmäßig deaktiviert. Beachten Sie, dass ein Bereich, auch wenn er aktiviert ist, get nur für Objekte funktioniert, die mit Algorithmen verschlüsselt wurden, die von der `cryptoMode` Einstellung des Clients unterstützt werden. Weitere Informationen finden Sie [CryptoRangeGetMode](#) in der AWS SDK für Java API-Referenz.

Wenn Sie den verwenden möchten, Amazon S3 TransferManager um mehrteilige Downloads verschlüsselter Amazon S3 Objekte mithilfe des V2-Verschlüsselungsclients durchzuführen, müssen Sie zuerst die `rangeGetMode` Einstellung auf dem V2-Verschlüsselungsclient aktivieren.

Der folgende Codeausschnitt zeigt, wie der V2-Client für die Ausführung eines Ranges konfiguriert wird. get

```
// Allows range gets using AES/CTR, for V2 encrypted objects only  
CryptoConfigurationV2 cryptoConfiguration = new CryptoConfigurationV2()  
    .withRangeGetMode(CryptoRangeGetMode.ALL);  
  
// Allows range gets using AES/CTR and AES/CBC, for V1 and V2 objects  
CryptoConfigurationV2 cryptoConfiguration = new CryptoConfigurationV2()  
    .withCryptoMode(CryptoMode.AuthenticatedEncryption)  
    .withRangeGetMode(CryptoRangeGetMode.ALL);
```

# OpenPGP-Schlüssel für den AWS SDK für Java

Alle öffentlich verfügbaren Maven-Artefakte für AWS SDK für 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	AC10x 07B386692DADD
Typ	RSA
Größe	4096/4096
Erstellt	2016-06-30
Läuft ab	2026-09-27
Benutzer-ID	AWS SDKs und Tools < @amazon .com> aws-dr-tools
Schlüssel-Fingerabdruck	FEB9 209F 2F2F 3F46 6484 1E55 0 7B38 692 HINZUFÜGEN AC1

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-----
```

```
Comment: Hostname:
```

```
Version: Hockeypuck 2.2
```

```
xsFNBFd1gAUBEACqbmFbxdJgz11D7wr1skQA1LLuSAC4p8ny9u/D2zLR8Ynk3Yz
mzJuQ+Kfjne2t+xTDex6MPJ1MYp0viSwsX2psgvdmeyUpW9ap0lrThNYkc+W5fRc
buFehfbi9LSATZGJi8RG0sCCr5FsYVz0gEk85M2+PeM24cXhQIOZtQUjswX/pdk/
KduGtZASqNAYLKR0mRODzUuaokLPo24pfm9bnr1RnRtw5ktPAA5bM9ZZaGKriej
kT21PffBbjp8F5AZvmGLtNm2Cmg4FKBvI04SQjy2jjrQ3wBzi5Lc9HTxDuHK/rtV
```

u6PewUe2WP1nx1XenhMZU1UK4YoSB9E9StQ2VxQiySLHSdxR7Ma4WgYdVLn9b0ie  
nj3QxLuQ1ZUKF79ES6JaM4t0z1gGcQeU1+UklgjFLuKwmzWRdEIFFxMyvH6qgKnd  
U+DioH5mcUwhwffAAsuIJyAdMIEUYh7IfzJJXQf+ff+XfOC16byOJFWrIGQkAzMu  
CEvaCfwtHC2Lpzo33/WRFeMAuzzd0QJ4uz4xFFvaS0SZHMLHWI9YV/+Pea3X99Ms  
0Nlek/LolAJh67MynHeVB0HKrq+fLuoRwepQivctzN6Y1N0kx5naTPGGaKWK7G2q  
TbcY5SMnkIWfLFSougj0Fvmjczq8iZRwYxWA+i+LQvsR9WEXEiQffIWRoQARAQAB  
zSxBV1MgU0RLcyBhbmQgVG9vbHMgPGF3cy1kci10b29sc0BhbWF6b24uY29tPsLB  
lAQTAQoAPgIbAwULCQgHAwUVCgkICwUWAgMBAAIeAQIXgBYhBP65IJ8vLz9GZIQe  
VawQezhmktrdBQJo12ZrBQkTQxnmAAoJEKwQezhmktrdi18P/A3De83MBx8bdcWJ  
Fot71Vk1TyBQFErgtrcytSU0czEHx3tGbZgQLbMlyzjir0T03usxEk0eqTVK+RU+  
5uFXNZYQLwMj1HJ6S8tnfLe/ExM5WQ2KPwIUPfZs1GDDRQB2dIKSc+qYrP101vf4  
04iPgflHMW2bFh3zjxcaHCJyqc7Cau33eZFBAsRni1j0Uo7MeyX0h1XfW8pd48Q  
wZ1lQVZ/6KmDiFWA0CZ+2svJ5cL0tgPoh10Qjoz0nHpNfuDILMrZ+e7tx2VTlkGH  
UGeNSydnrK8v9ztFn34KtU/k7NEWoVSyEi+5ICZL18FBwPqTwdVWXwXrqZCKiIpr  
8ZdJWdz2sJfgDFNCC6rKgCQ6FirmaD9G76dYwkQ4AbZqAB1UzU3q36W1K0r3i0Ab5  
G4td0t4yqXHTe1x+ZUNaeW7gaCmtXAxLw00feJrcq/44b/SQP+qJ8sS0v76Yg2oF  
BsF5DW0VUFghbTyokHAoVR0yhBR4dUUisY39AqLSL8+Lp9Pr3wNuG19GLrMD5701  
piUb88B3Gwe1EiKV1gaKrvZ3mECDUiSMV00Z5iG8E4QDpNmVbJbV1uT821ubvt0v  
2Ko10Fa0uWcYgssdRGqEXNy6jz/Er8LAC3+nmGINDJQzrF+loYoSSkI2Nu71hMuL  
7iWwUPF70hDXoVSan4X3x6q2rGK0wsGUBBMBcGA+AhsDBQsJCACDBRUKCQgLBRYC  
AwEAAh4BAheAFiEE/irkgny8vP0ZkhB5VrBB70GaS2t0FamjXZTsFCRNDGLYACgkQ  
rBB70GaS2t0/0w//YIv51vHtD+kwMmIvk3zpzizDHY0zW2d0ezAo+C/DsSyC7wDl1  
Dixw34EQ1yLXH5xLR8CH1zup13JmmEp1ucdQggoefbidxD18F1d7tJ0D1y3GGnTD  
0jAl2ZC+W650h+wS1mD1FlaKjMGgkvJf0dA7RtU2T8dv3vt8dsxg76FMFS3+fq1C  
FN0AsNTn9zWR1SqbIfkMJK83aq6s/rcEV9VrAYgDgqex58fygB5EuTf842/IF7WZ  
Q9gd6fupB0mMzP5Ywd2uj/vsBTYakG+mgQwDxZuKPeEzAqnqqS7biSQ0U06Wozlq  
Yy4fSczE9GkBAvg0pGmbko+zHvpnjvX/h1CupC6odvFy0AhZp6zyhs0QWz9thfqV  
lU8W1bgJ2atFDn5GUSxF/fe0Yzovlbb56sbYXuvMG9RiE0uJ1mBbZR3aIdZ1U6Do  
BHc/vjc5mWcV7JQSP7i4W/8W7X3UAuN9LdxB+IvF3Cwrgtlw2BWvA5A1co5Tnz8t  
P/CIVmBjk+sLme8W4kfLK3IWEbwCl0dNnErI/MHRm65A2Y5EMihwjr0i07SU1Pxa  
nPPg30YJCdvjzdB8QE3/DBiMf014dISfKDVEWnfK8mZaYd/BeRm2gUAa9UrqSFCG  
BlA7Lg+eLI3US0FvWwJ4j5bBJqgLu+y7crIkiU0PAQuLk3l0+5uYU/I3DuLCwZQE  
EwEKAD4CGwMFCwkIBwMFFQoJCAFFgIDAQACHgECF4AWIQT+uSCfLy8/RmSEH1Ws  
EHs4ZpLa3QUcZwAXCAUJEWvKgwAKCRCsEHs4ZpLa3ZdTEACMBLg2q9zk8ZH02nDz  
Sg5zc8Wlqq8WdxU0Pj8qx4U0rrMca7wyiUvrgoxPW5l1hRVNUeMkDRfu9pSxc0VI  
V9LvmYE/WnwKR0ubgGbsC4T7M/LqV0/AuLXil4d7IXc06l4toa8LTNwtD5b0DgrN  
gvay1AzCU8kq1Qw1cKZ2gAfvA3Ba7PWYLeUN4HTlGrXcw73G+0CofY1L8wqWxHCJ  
29XqQzeTEc6MDEeIlNlVdUcy8Qr5uwkEs134H9AxS5F1opJ4TqvXiDZsrSRRv57R  
XYmRZDWeYT+9PZaMsHXza5qgej7BfATxhYfICsNaY6MK3x6b+nDSKkoZg0+i09zh  
1YjppahhQe6G336v/3mRj0dKGCRCQ6znQ9ghUaB5z9zfvgh5A0EkTe3l8MqM+j5A6P  
VjSBBJAHKEjxr7+wKJKIA6P+DqpsYAunzftwUzrLVqb+BZQ+DcTmVrE70PcMYJD5  
Qg1X/Le+WmWZHI154NXgpWU0UgZUBUge4DKrT+zCJ9iecPLKTW70cULyX0+rjb8  
8BGriD5GP1HB3d0UXXTlMKCqg3qy1Bu2KnZTQiaEEedZgSIGQbrW0JTMmmXJkKjokd  
JMA4vYeg5en51G9nRQjScPngx77IxxvByNyFwTJdG1ENpJpsK9TtmENcpyUJtJZTJ

ZS0IRVPP5RzR5vInuXWq6VV0BMLB1AQTAQoAPgIbAwULCQgHAwUVCgkICwUWAgMB  
AAIeAQIXgBYhBP65IJ8vLz9GZIQeVawQezhmktrdBQJLJEoiBQkPj/2dAAoJEKwQ  
ezhmktrdx1YP/0vvym3jgX/pwnR7K1rafZMb1iKQBr0ISG8cdbaf4ppX5vuUZnyj  
w9Cl/o0Nn7jJjnQx0IIzuBoxne2WN28ftM2w0nVxm85mAmz2fwQz/fdKDyonXc0h  
pFD2iMqn7gESjhEgRE7wMDYMDuLdqHI70KWGVfgrh7xEmKapLh45h7cnumo2VjL9  
uDYY1a0BHz993T7oE41y43rhk+6kKbGFd2uu07h5j1ZF8Lj6sYfcEzX0U10hR1D0  
nyBjDy9MYWu0YNouc70WgMceGx6hjvCAM/5fxP7SZFecZ7ePeB0GpvVA24hSNENE  
0r3tUeku0f1I0FunMnMnbh7Z09rPYqWvWdNIpU3S4CjFhY82L+IeKnmLy8N6ASRk  
HsPiNCOHSK8C/0ynrd9xLhX8Jsk/TGiQYaleoHhWkNL1ZsL86QHL8SKEqkqZCQf5  
AEqghDP6NEGS71n0enA7JjIrA9KL1T7fnNWZ0wFi5X+o/CymE2ytEMS0Yf3nmY4U  
n9x56Wgn6J2zqB5nq0Xf6NxDGAIg0Bm098YEnKCIFzk+yhoDlprVpHcnd2b5f60q  
uh8KY0EbKgpMJ3zZuW5L5kwGF1nNoYiAkonMaz9H3p0Qn0MVYCUeUTDRsi0/prrd  
Uhn1ry4TASBmpeXnFhdLVM3vFQZVpByadG0JNmnaN/Wavw2a00UGBFa4wsF9BBMB  
CgAnAhsDBQsJCACDBRUKCQgLBRYCAwEAAh4BAheABQJhMqGaBQkLn1UVAaOJEKwQ  
ezhmktrd2sQP/3YHM+U+Bb0y1nSEAFykZ71+uCM2hkHMLdxQYWB/rBwkmg/pbu+d  
r4t45RsTASrNjRcZ0nt1PMQRiQ973ymHfpmes+noFwvTGH7zDv1BRBR9wPrd1XUz  
iSuEUHGi/fqxUVXQ5mbonzfThX8tuXeuIQmeToqB00FY1Zm6xsNnEHcjV166mC4  
IPoJLWnZJs4r0CeoRf5XvDTgX6xt5/kLYRZf79qaWGFvaZpsc1CH+rQJUdVa/D4T  
7pI7hX6zy0S91z4iuC5HZUi0TF+y5auEZHGtdTWNS1kv0vfcCTi0XK/GkGL82SZu  
7X2VGnpCeUnFyViRGlk+KaDG1sVyDY+lcBPg6ilr45M6MQV0iHS50F04QNXSkt5+  
UnzJH71ldgNsR6ibRMyNV3k5v3fyUcSbvIYyLORTTBiVEjQDSbk1QNqbrQ1X9CWz  
+EJWn16BFTmMFvxBSWpM640GncHP5J3/0MbMw3Cm90x7k8UfNANIemcrJrSxIDwm  
g9cVAg3a+D+wxjrVe8jGg0ejvECpm+0yswigj5x6Lqj09A4UgdjEauN+/pn0nhBo  
Gv7DzMXtM/LoDtgp6wn93qZVN2TsuHnkEk4UyntB6eWJbBdXHWUr47exiWh0dvQN  
tpwCWPT6I7ZTPtA5K/zx+q9m6797BLgAkTYc6g1oQL3vs1Z1S3m/hZNawsF9BBMB  
CgAnAhsDBQsJCACDBRUKCQgLBRYCAwEAAh4BAheABQJgmrz2BQkLBnBxAAOJEKwQ  
ezhmktrd36oP/2rB2EkwSOCKC4m0heWSfDWi60BkoEbbDtFtc6/HwqBW8SPsiK1q  
zV0e3qBY/LVju04+ktJEK+EGXlnC3iC36MegrQ8zt391kEx/Zv9LIuV0CX90QIAX  
dL8MVUkkjRLCFFH8pTgRy1cJYwk1X4dYdXWYc29fCwNVartNdNBhsb2ht3VJeKDE  
kUivBhmkuISDPEnI1coY7Lj0ZtY5cHdRF2eZpB0RkTBpsIt18rCYyHkerZrhmb  
j3r0yPyv0a+1/dQs8/hv5pEmbKx8cy8RdJkmbUHYatPBsjHkJSWr707G9VFW4GoN  
9CRAI4KkbDSEDjCL5dv2pq0Sew1MkLuWJGULAMgiIU1Wc0s5SZZGFSksNQrtSFV9  
Z/wGocecMgkGQNXQ06JV/Fry/TvyphBlmy1EqL+NLqEcEjn1z90IVu+ZA+M09J96  
ULH07V5GvBgM+QK/q/dJeMHPWrNlo1gA6NwL/HBdM0DqzdZ2jEPvsQSABvZrPMty  
+BAqEar4wqY1AH4X5ccEj07nJQoBQSDRSki1fkBsc1nx44N/m0kHdIa0Z/Y+Mw4v  
WiZhRek0ospG1I41Ba3CNTVAhSs9msGsYfkqvFJGHL7sZY8XSv82GBBvA0nUNrsJ  
bLBwo2FaQG9eoatRAGkqp4b/0tNtBuGeiQoNwFGbfUZTAaStj5/zZj0sWSF9BBMB  
CgAnAhsDBQsJCACDBRUKCQgLBRYCAwEAAh4BAheABQJe+9bwBQkJZ4p1AAOJEKwQ  
ezhmktrd+ScP/RoaUKriVvAgLH0Gs+/mnfKtnfT1Clzi5dsdI9/6H0vLpmSWK/C1  
2cT6gary45VMgAeVK+H11QXafYj+FY++I5kYoe2GrSvIXhpjaFAJyNf/dK1eTsqR  
Tm371i8b3FDYs5kvy2CnTbmHB8Ms0Gxck8/YHd1x+g8Wp02Igf89yYCSF3CAdx3  
6bHbs6Z3C31cM/3SoWF+Yie2P8XeBMPGp/BcjQzUcHF6G06TwDDYhixucUi6vEY  
EH5Jt0wVVQ7bubT80Fe0oJwVxlzYz4UoqxjKDWymarTzu03AUIT0PXPece94bJAK  
mSh68ItQe3H8tSPMubERWz2tEV31VlKChDGXcC7BYQmxHseolxz/qzCtJ0iX9BvZR

dniZNeNJ/Cu8M2pDp47zdNFXzf/Q/sQ9pQ1ws22G2g119rWDneBku9n1vTP80/er  
SB+VLTBjDiArLCY5y9+BG8wbscExJySoQxkB9j/nlMzPY5rgk0SyxsNj9GbqH+hr  
EjS3/uacNwSLxGcOT2E9Teot5pfTE06fQVq+35QhfA1P8c8jze01W/+u+wXu1Ui9  
azRSzYtCHanGyyet6U1mlBpAkqkZzH6t3CA5czc9i6FbzjvFVZnbRUZIRzfISYew  
lF5WqgTn2iYVdxagPRvLF5kjD696brGW9d5HwirCVGaK04VsXWlAb1B9wsF9BBMB  
CgAnBQJXdYAFaHsDBQkHhh+ABQsJCAcDBRUKCQgLBRYCAwEAAh4BAheAAAoJEKwQ  
ezhmktrdWigP/3QWl7a081BUWyby4HEhN4SdAoWGY/FLq04mCtup1cnMgRUCSiL9  
l2BSCTMctUcdSwTtYw0gSChN2mMsd1U2FNR5HvNunYR/pFdqjFQurflZmKVeG5/4  
uuKa0xMw9e8pK5uYAFs+07gr8gu/f6/Drp7NZk3/yVKpf4WCY9oX9TA1q90/11nN  
cwS45U/d7YP+N1YM9cBXa1DnDcdm0BlykzouAF0qd1Lwi/tmLENvybD3+2c2WsE  
r1FZGSa5Zaf00tTIWxh5k6wh5FdRRycrnSyRK3B9N9+yaXfMQ0Xp0ypa8dqQEnCi  
IsngDCJPxtTrhMwKhBFRUMzK/WZTDboTQSQDK+YVRrE4K8MtoZSKwZLV2r903TpX  
kpbKsPVYmexerfdMeZfjZMF1bC7BmEs7jciH6JjbqAoAPnHzN0481aeNarINSViX  
PQWr2mp9qShei2/RavLtx2ZNRvmGW72ZKpF8E3WUdPBJqFVeGNRv0m3aZj8o/Hl  
ewtNjcT4ouJfq1fKiULv+g7ANEMDLQTFDTg5twRdvmZ1B7oTBSavf+LwxPIXhH32  
IR7TX7VeicMMxmZnmZK2ANT/QBi3laf+ojVHvB+f6D74eLNq0Zqjfi/3UFNysYjg  
E+YgCqEUBpHb161n0HwG0SsQwfap2uKK1zukD/KxH5SPBC3DYGBI+KCbzsFNBFd1  
gAUBEAC8zNArPwb3dPMThL2xAY+fs60vXdB1Sk0tYJpDwPfgvo0d+VQ+hV6XuLGA  
HAS6xG1WHysPT9KejIRSgLG+e9CaM5yhsxNa1WFGUM4Q9ESo3t+a75Go7xHIxgFj  
C046/06Vh3g9N/PREeuG8zkZ3H2v5fmD+ejyPgk4W9sFL00zjRiZD0FKVYR/j9ue  
nEC/2NBcLuFy3q6CdFmCoDE0062kXmNaGz3knzEK/X1SkcjsxRDq7zaQ1Q1Kou+3  
dICwy4x5SjQ8j1+eeeEvF2C2/dXmDohb57tqUwioohMUQkmCtvZgEHjypUwgp0MT  
o25gWxkvJ1SJKU0b6b1786WnySIzF2gxq1kkEmB14RAssQkeXjrSmGwsMDyHNqyJ  
eYFusl8sPaSpo+V2n0z+2B070Uq+wmf1S5A5FpegH0PZzzoNZo8I6QxaZje9YSZU  
ijGmZIdEBleRVt3Svhi8MY1nasd4bW2RK1sr7p1kBf8QRe6biiQRF3KD0Sn5CbmX  
pAchJ1ZHzRRdkXZDNQC6vCJxsy1300TrhJtAV1Yq347uyUbVi291ISVgroUVtprs  
mHoEk5Go0THbg9SCSt+xi/FiJQC+ubWmIGXoFKMR3UmhDnnzobKcbnbs/Hd981Fd  
VghYYvq//gTakJk0WxfGq030wtXRndPOA0T+qhP3TE+LtGRJ+wARAQABwsF8BBgB  
CgAmAhsMFiEE/rkgn9vP0ZkhB5VrBB70GaS2t0FamjXZm4FCRNDGegACgkQrBB7  
0GaS2t3y5g/7BFXp/fdanzuQPToJTPen7AVwhLloKaiYhG3GjdXfMPLvu6UtaaGm  
qynLo1UNNoobptFqc1G9BKoAghQrta7CsDhtsQF2xyc3Mfu0gmpL/7X5a7sFIeJ  
j08UjfwHx4DSG4LEZgNaAoWFjZltp4+8cqijkAHxt+r+1ayQG4VVH0WyXXqmSH4  
9HqtbPcPyRzxdVLeshZC9jmhHhhKqw/LwGyipWSOUKQDjWarBwdyhNmWCaLvXh1  
ndMp4tq8DPGC3G4T9tYAbANrn7nKfZgHbMSzMw9kSp0L6QvwwTDjJyIWz85WyeH  
WHeBysDaB0it3XD1ehUew27y7N6a9hQSYjnXuwwre5mjDI0qJon/31R6ui2Z1y9P  
a+bC11hbLXXh9tLCXRuo0t6thh9Cq5X1a76PPpEv30o3bpsb6l2hbrut10KezvwK  
l7txito/jfMiWfsZHA904SoM+8GnmVingHtZ805n1T4RddJvT/vaqp1fI6zf7jmf  
a69lALP420riF0QcwntNUM5tVmFUZsnFp2YRd4Ls7MiXVjtABahLSbb94l5WSVc0  
jr0LDf94edvzk4R8i20b8CfVZNqEsTR6bHz8dT7Q+xQzEdjUujyyZY1UU1157Qeb  
0sHjhCtuZYCI04X9hZ37nKnZXSxR1RDCnt5BEiyFu2WD1RscUe6PcVDCwXwEGAEK  
ACYCGwwWIQT+uSCfLy8/RmSEH1WsEHS4ZpLa3QUcAND1PQUJE0MYuAAKCRCSHS4  
ZpLa3XCpD/42DrcveE+q2ulrAIYDPD1ULHiwIMEjqBDRm6zmr1KSAeb4E6/MFcP4s  
rXSSscMlrqG6NVynjNCXjD2YzWii68EwoXLJkgoD3r2ifzkV62EX2MIEeNZAVwuy  
KNxorzmy6bhuWltRYNK/hITs2AG5or0k9ADEJ8PixKymrWlhesPaWX6Yhp9/tWaC

RHOSRiLbRVaJ+7sqT88urLmkV9Hqx949Zxv4+cgBVUGL6WXXsfWhHjbDMNJnozWB  
SZaIJznLAP0M8z+1DNrUyYfr8SkF4IOVmg6HDzoyuseJJ8JvMA1kvT6F9VBq/iE  
yeDYdEEQxwHwozKriEx5Ybx15mntbqwCXY6kHSx2+/3RZWPZQ8K29YP9QEk0KeGF8  
9Vap3jjNrx4u3cuRNQpeblQc4uFn3Nzaj+cVV4YzcRw94NifecXpujSvk8XU2ytJ  
/JgMBxPIBKglN4eEMet9b4FRB5XeBdPAm19/LXyb4I1IipGNXlgNz/HCuBzidzHT  
QQdqfA9rZVx1hwFr7AJCVqWaXVsx1oEAhKqTtsLMyj594DvnRuwKw5Vse+1eydW  
MIHYdbxmJccsTGIIt/hs0pc8zfm+QYk5752jshh0KEBy+Ey3QZI1Wb0547N0b2Hwr  
Pgt7fw2NCKMPE1Su98zmeFPhqNHf7L5urBe5gADj81E8lm6t/oVxcLBfAQYAQoA  
JgIbDBYhBP65IJ8vLz9GZIQeVawQezhmktrdBQJnABcLBQkRa8qFAAoJEKwQezhm  
ktrde3MP/13CLWp99XvRR0rzD/bW0fWjAenT2PE/tYd0Y9YcTQFbnIUhaVUDWAo3  
pibR3D4u9L1Y4o1pGfJ7BTIHF9myfpaVvmrNjueYI4omli24JQ/CKqNdY8Qzxxz+  
/QyiNK7Aw5cEBWiu84WGB1SsefWWT3rZe9YBb77gNcWHZ15pXTXrcgUxGY4808MC  
I9YFWq8EA0iHawtFnmB3UFfC1Wt37Hy3PKvr1is3uG60+ULI8RQz3/+ZwSG8U+xt  
b+I7H9+gITc1eFCb+tIwp5xWflyxcFXyk6Uz0L7y3Fg2tIEuSNtIHUC9NDVobf6c  
I0KAzZcMvKiPQiuBnV0jgDLmCZM5H6axj9x+gi4oVh6ea3HLqMzyjm5JkeCGgKwv  
H0gD3yGEZDVcbavkQ0le5T+4JefndKzCPrLuX0iyx+oQii0L8WieSSkSB6BsZcUN  
SeuGJwM79Y70qlD/YVrQNBZj5Vz+m3nZ+0EWDMMI0hRgMpSEIc+dnTC0u103Z+Rc  
c2IJq8INmU653sUcfCZE12ParW4rF7ib6kViYrABT8f4e2TP0a0yP5kp51ied9qL  
azaBA6tt/C9X1V2EJZK4srXtmcZ02Im45RAiVXyfpBAmmiF3eZWcbKe7qBC4rDRh  
LZG4RQW/S86Da0BID7gQz9IFSkaG504MsDhvnA7iAqaHUHUpCsiwsF8BBgBCgAm  
AhsMFiEE/irkgny8vP0ZkhB5VrBB70GaS2t0FAMUkSiQFCQ+P/Z8ACgkQrBB70GaS  
2t3AwA/9GkXKUGvjKGCxwE4SdDt7c2jw6to2TTP9iFJ3Xbk3+5BURT3gkZCuu9D7  
gt+97aVo/B4EM7Xz8DQKyY7Ic9VAwDRra/Hwi1V0hw1zyIWQ/gAnX3baU6qLRWHR  
vVR5meV8r35C+rg9DaWfYmvS7PIv9LfxESwBPUjbmX8k4/5EJpHUwf12bzkTnot5  
7q51HxKQa6IvqQak+Hp9ZM2KpdsGK02HWJJIIvYcI5byW9zBKV007YR8gtRAJKp9  
IbtsXx0WT6cqH0FVc5SSzdcaMt0gLF17BTnJyvKK219GABGBmzYDjeCyF2J+Ippf  
oqxqfTe6Eo0suEMc2PbLTs9SswjyCC2VG1X8+uUH9SoKwL0VQ6LfsP6fhkVKqi/a  
rB6UuPR/iZnrKIuxMNQ4U+t2Q6UdMlMxsAXTNDkwzok9oJRokIrH0ZV1KtH4sJJ  
tCic+t0ddq+GQLiKe2WpJfX1A0uESCB0TxjAwQmfn1H+dUhpELlBnimH1H0/hXPd  
ifuNGozzADIRseQDyZjl8xGL1qRZLD3cfmda6RyZ+S3dQRuaRrcFCDccpY/p0+F8  
jbx64zyqqNs+KV+SkQG0cKFhWTZGCfQ/zMDtDmQKjb3eTAkv1zdE0Mw9zEjJmS0q  
8FNl+2w03VnvXwvBbtDdVCIaIq+jVcsy5XtnnV+bJ19Q9yue/XvCwWUEGAEKAA8C  
GwwFAMeyoZoFCQueVRUACgkQrBB70GaS2t1uHBAAh0YVvrtchRmzCvdNER1DtkIs  
bgQPJ90xbyfvmvoD06qxH7PrycLZKbt7yYpAUU/CMc86GwaEe0I5Nm1CTs6NvDlv  
g3e7EPIS859tyQflbM56N1wbsopCuoCJYknuoIf/M6dW6vJKNXLMmnL/AtalUBw  
X+5pb1mGUUJep49oT0xQEnvnuqyvaGjXgFXix5PVFJD2ed5NnQeFpvcCpc/ioN0j  
z7OR082j1ht5nWqPraXX5AYhQFM/kwR1cK4LV7gVDd/q+dfGYHzpxQ/HtyX/Lasi  
N6I52QqA95SM1ZZLPFLaNH6EvnB7uC9pLCYS8nvi1X7/cez5Pffff1e1gXCOT0jv3  
mJ2exLmXV0BbfKgjccFCxhrdRLtukfiDfJkySy1zdsncpfng8wJ3xKRv43cUTz7M  
Z240YNMqK26aJZVXEQUYjCwsBy1Y/F5wjYAWgwZ8yF5Rfix28P/K8JsIHb3QrAJK  
sNWQAb03ZWis3N3spR5M9Mw3VuDZ3WUXq7mxB5M3kpVoZ3vETU5cwTbADYNP4Sw  
BDK2uIVtxabezxBtZ0FcyYoF+0W8q7r4WvoyC9/+3Gfn0zZLJcEIVDk4W2pMW4A  
UhG/6drKTm3HkSDWIDu7d1sHWMffLEYfUhtN5DKkDkGoPfhvZvu9teR5yLfuRPTf  
ktihPn/JMrmwa9pwi8LCwWUEGAEKAA8CGwwFAMCavPcFCQsGcHIACgkQrBB70GaS

```
2t0uaA//UWRaRiHEAKeRqBG/T2ak+XZJNu7QHfNgoUEAub9Zru8oPPXx2AJLcHEN
KWmeFLLxAdDw0Zs4Bm9o0ew3VQnR/dBqjnXfob9Rc+eYUjA3rXazM/QrqcU8Syi3
MjNGUmjdL5aQF+IppAMg0BLG1TEmM7C5/PvrGJuYpGEnkKEwMK/GYhqg2V60pHEV
Pvs66mefJpCzbZSy56qtknSt6yBNWc14XgDX6VTn2kW4CV/3vVJUuvjvYs9SPyY8
mKEXa6QvUd3PcXv6RiWk4lGYuT1+jh2VkcFQ+JnUwv9TbKFB9b5jq1bvW9+LMDEl
YXux7pBP5Rpk+0LpyiExIRFWhi3x7aMW0zQ+I9yuNTEYkTHiEAQRUhs/1Fh4oLgI
v9QZgC0mRSN3zm8plQdivs1Z1AosAqqkA9BQwqsgosQe7P92irYIJqay0si9wGCD
wSMsmeXdIF6wW3/UMJZL66aarPeiZApGX0QdTzwjMh/QK/8gTKyeZulKmNkNfwWq
0170irWqLKssVHTg3VUM8EIdh+oNqDDXSeWtYUmpPpWp+yWZ0x1MFFZhuQHQTGu
TIj4A92LQzbrfj/jXrVWm2SrJMivUoiDUn+qxKIvVwFlI5gVb+uyTFhw89Pckphr
JwRi052RLou9yd6Ek46UH4XfZZWrZuzY+zzB7oqG0NphLgi/h3DCwWUEGAEKAA8C
GwwFAL771b8FCQlniTUACgkQrBB70GaS2t2/MxAAjoEGPdzavhs0lXdPCRd1D5QJ
r8T/NSEV2z1cp8ZvdrkjNF09TBP4qsBnKJiuvY1Iw70GX9W2okvXxgJizE45v9MH
WEMz4hmIjmAfRwcqENgp0c1IY/T0/+kkCW8dB6d30J1kT0n2PCRzN9L5vPqZXGTG
mLvd9M0jH1256w4uxLb+e1HMDTCqEN1ppq9G+EAR/29q8JZWs1marbZZWxSWcg/E
1YYbNafzklgjq4CLh/j8AEWSvLr39zRy9uvQ/yqAKZ4K4aZfh/SPupGDvsD6ZK54
EPHxerQ7aiXTbUHTvwhxWLOP6WmxFA3Shr6L6YUb6jq+0PVliFC517g3mxFHJtw
yXGNiKhzmzr0190lsHafuLJ/9QPfK3Ce32SkPhW/11MYA8HzduMv5Arp7cBczXSP
EUTmNIVKv3gTjSQrzRhwhHmMuqyDZ/rXQQ1j12sxIDj04MUMvVjYKF+OCNm42gVs
8ca3/wN9ZNU6hyFWeKQDuCAqPPbT5G0/DKseFEwB+07wwyH1RXbyl0v4fneg605X
S71qhNtw2p1hDL0HYHDiV+aPZ+LoOmX6+dmnqE6bQJaIlVb922KwmlI07F3DkqP7
0jFlhoE1gfiXWkxP4Gy8w0obNfEMgvz02djKQy+oQqeNdIcZfZgzPTGKB/nVgpt
9CcRDWjPltFCd2e1FBbCwWUEGAEKAA8FALd1gAUCGwwFCQeGH4AACgkQrBB70GaS
2t1PIQ//Qc5VYfBCxpaMysaPQ44wXPEZSjxIGZhhMGzb1UzzAEY0w+RgKN5nNTXq
L2Ko0k0rGnKqZ0KByMdXwIPH/rGwwEsbbIpopnibf5ic5B/+xCTIK+qLIwX2ZLuk
NhbL6Y+E+7DxMMH+KqBWH0NkkgwVY+rFW0foops839ABKvc9/Ry4/qqkcb40AzpD
11iQJ5vK/DMuaDWxWeKXqJLI13WMGPcPfheuBZL1u7LEEHYKMgzvpbf81WIn3MBo
8jvxf2/o+kMafSSDqgv0u6yu8G0hmScpCbRjN7jV/HrG+tM+zy48TN6/MkGWSR7q
TD34pqBjyatVfV16dGD6xj/i/Emt5hZB6qXruCDH7AWMoNx+FkDubs4sc4PKysZU
Itya6KdQFo2UeYsNwZhdn6QwKhd85um4JUHJCY0mARvjsQgWXH/5MR40cow77bbE
vVq0XNd+QRVlyT42CEtnIUOFLedVuzrum5Tuvvna6ImMDoi/z6QcNeL79XsY2m6I
QVRiHr1BDdb/8JLkfnWiwL8GRv169Kf8unx0y5u1YBpcMYkyDD2+pnnk3TY0rR+8X
8goecaS8fbyu/Q48K85ZMD8wKW/bzLQ+tK9y8xed24u2QERftMhIw9b6f45Nrrf/
PhgV8RnuwUusSbdDe8kw3eYtmLdzD4kZc9K7Sd02CqT+hm//9JI=
=uGHC
-----END PGP PUBLIC KEY BLOCK-----
```

## Frühere Schlüssel

### Important

Neue Schlüssel werden erstellt, bevor die vorherigen ablaufen. Daher kann zu einem bestimmten Zeitpunkt mehr als ein Schlüssel gültig sein. Schlüssel werden ab dem Tag ihrer Erstellung zum Signieren von Artefakten verwendet. Verwenden Sie daher den zuletzt ausgegebenen Schlüssel, wenn sich die Gültigkeitsdauer von Schlüsseln überschneidet.

Haltbarkeitsdatum: 04.10.2025

Schlüssel-ID	0 x AC1 07B386692DADD
Typ	RSA
Größe	4096/4096
Erstellt	2016-06-30
Ablaufdatum	2025-10-04
Benutzer-ID	AWS SDKs und Tools < @amazon .com> aws-dr-tools
Schlüssel-Fingerabdruck	FEB9 209F 2F2F 3F46 6484 1E55 0 7B38 692 HINZUFÜGEN AC1

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-----
```

```
Comment: Hostname:
```

```
Version: Hockey puck 2.2
```

```
xsFNBFd1gAUBEACqbmFbxdJgz11D7wr1skQA1LLuSAC4p8ny9u/D2zLR8Ynk3Yz
mzJuQ+Kfjne2t+xTDex6MPJ1MYp0viSWsX2psgvdmeyUpW9ap01rThNYkc+W5fRc
buFehfbi9LSATZGJi8RG0sCCr5FsYVz0gEk85M2+PeM24cXhQIOZtQUjswX/pdk/
KduGtZASqNAYLKR0mRODzUuaokLPo24pfm9bnr1RnRtw5ktPAA5bM9ZZaGKriej
```

kT21PffBbjp8F5AZvmGLtNm2Cmg4FKBvI04SQjy2jjrQ3wBzi5Lc9HTxDuHK/rtV  
u6PewUe2WPlnxlXenMZU1UK4YoSB9E9StQ2VxQiySLHSdxR7Ma4WgYdVLn9b0ie  
nj3QxLuQ1ZUKF79ES6JaM4t0z1gGcQeU1+Uk1gjFLuKwmzWRdEIFfxMyvH6qgKnd  
U+DioH5mcUwhwffAAsuIJyAdMIEUYh7IfzJJXQf+fF+Xf0C16by0JFWrIGQkAzMu  
CEvaCfwtHC2Lpzo33/WRFemaUzzd0QJ4uz4xFFvaS0SZHMLHWI9YV/+Pea3X99Ms  
0Nlek/LolAJh67MynHeVB0HKrq+fluorWepQivctzN6Y1N0kx5naTPGGaKWK7G2q  
TbcY5SMnkIwFLFSougj0Fvmjczq8iZRwYxWA+i+LQvsR9WEXEiQffIWRoQARAQAB  
zSxBV1MgU0RLcyBhbmQgVG9vbHMgPGF3cy1kci10b29sc0BhbWF6b24uY29tPsLB  
1AQTAQoAPgIbAwULCQgHAwUVCgkICwUWAQMBAAIEAQIXgBYhBP65IJ8vLz9GZIQe  
VawQezhmktRdBQJnAbcIBQkRa8qDAAoJEKwQezhmktRdl1MQAIwEuDar30TxkfTa  
cPNKDNzxaWqrxZ3FTQ+PyrHhQ6usxxrvDKJS+uCjE9bmWHVFU1R4yQNF+721Jdw  
5UhX0u+ZgT9afApE65uAZuwLhPsz8upXT8C6VeKXh3shdw7qXi2hrwtM1a0Pls40  
Cs2C9rLUDMJTySrVDDVwpnaAB+8DcFrs9bIt5Q3gd0UatdzDvcB7QKh9jUvzCpbE  
cInb1epDN5MRzowMR4iU2VV1RzLxCvm7CQSyXfgf0DFLkXWiknh0q9eINmytJFG/  
ntFdiZFkNZ5hP709loywdfNrmqB6PsF8BPGFh8gKw1pJowrfHpv6cNIqShmA76LT  
30HVi0lqGFB7obffq//eZGPR0oYJFDr0dD2CFRoHnP3N++AfKA4SRN7eXwyoZ6Pk  
Do9WNIEEKAcP6PGvv7AokogDo/40qmxgC6fN+3BT0stWpv4F1D4Nx0ZWsTs49wxg  
kP1CCVf8t75aZZkcjXng1eClZZQ5SB1RtSB7gMqtP7MIn2J5w8spNbs5xQvJc76u  
NvzwEasPkY+UcHd05Rdd0UwoKqDerLUG7Yqd1NCJoQR1mBIgZButbQ1MyaZcmQq0  
iR0kwDi9h6Dl6fnUb2dFCNJw+eDHvsjG8HI3IVZMl0bUQ2kmmwr102YQ1ynJQm0l  
lMl1I4hFU8/1HNHm8ie5darpVXQEwsGUBBMCgA+AhsDBQsJCAcDBRUKCQgLBRYC  
AwEAAh4BAheAFiEE/rkgny8vP0ZkhB5VrBB70GaS2t0FAMUkSiIFCQ+P/Z0ACgkQ  
rBB70GaS2t3HVg//S+/Kbe0Bf+nCdHsrWtp9kxvWIpAGvQhIbxx1tp/impfm+5Rm  
fKPD0KX+g42fuMm0dDE4gj04GjGd7ZY3bx+0zbDSdVebzmYCbPZ/BDP990oPKidd  
w6G18PaIyqfuARK0ESBETvAwNgw04t2ocjs4pYZV+CuHvESYpquHjmHtye6ajZW  
Mv24NhjVo4EfP33dPugTjXLjeuGT7qQpsYV3a66juHmPVkXwuPqxh9wTnc5TU6FG  
UPSfIGMPL0xha7Rg2i5zvRaAxx4bHqG08IAz/1/E/tJkV5xnt494HQam9UDbiFI0  
Q0TSve1R6S45/UjQW6cycyduHtk72s9ipa9YM0ilTdlGkMWFjzYv4h4qeYvLw3oB  
JGQew+I0I4dIrwL/TKet33EUFfWmyT9MaJBhqV6geFaQ0uVmwvzPacvxIoSqSpkJ  
B/kASqCEM/o0QZLuWc56cDsmMisD0ouVPt+c1Zk7AWLlf6j8LKYTbK0QxLRh/eeZ  
jhSf3HnpaCfonb0oHmeo5d/o3EZ0AiA4GbT3xgScoIgx0T7KGg0WmtWkdyd3Zv1/  
o6q6Hwpg4RsQcKwnfNm5ZIVmTAYXWc2hiICSicxrP0fek5Cc4xVgJR5RMNGyI7+m  
ut1SE2WvLhMCwEy15ecWF0tUze8VB1WkHJp0Y4k2ado39Zq/DZrTRQYEVrjCwX0E  
EwEKACcCGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AFAMeyoZoFCQueVRUACgkQ  
rBB70GaS2t3axA//dgcz5T4Fs7LWdIQB/KRnvX64IzaGQcwt3FBhYH+sFaSaD+lu  
752vi3j1GxMBKs2NFxk6e2U8xBEir3vfKYd+mZ5L6egXC9MYfvM0/UFEFH3A+t3V  
dT0JK4RQcaL9+rFRVdDmZuifN90Ffy25d66JCZ50iqgHTQViVmbRgw2cQdyNWxRq  
YLgg+gktadmzis4J6hF/le8N0BfrG3n+QthF1/v2ppYYW9pmmxzUIf6tAlR1Vr8  
PhPukjuFfrPLRL3XPiK4Lkd1SI5MX7Llq4RkcZN1NY1LWS8699wJOLRcr8aQYvzZ  
Jm7tfZUaekJ5ScXJWJEaWT4poMbWxXINj6VwE+DqKwVjkzoxBXSIdLk4XThA1dIq  
3n5SfMkfuWV2A2xHqJtEzI1XeTm/d/JRxiG8hjIs5FNMGJUSNANJuTVA2putCVf0  
JbP4Q1afXoEV0YwW/EFJY+brjQadwc/knf/QxsZDcKb3THuTxR80A0h6ZysmtLEg  
PCaD1xUCDdr4P7DG0tV7yMaDR608QKmb7TKzCKCPnHouqPT0DhSB2MRq437+mfSe  
EGga/sPMxe0z8ug02CnrCf3ep1U3Z0y4eeQSThTKe0Hp5Y1sF1cdZSvj27GJaHR2

9A22nAJY9Pojt1M+0Dkr/PH6r2brv3sEuACRNhzqCWhAve+zVnVLeb+Fk1rCwX0E  
EwEKACcCGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AFAMcavPYFCQsGcHEACgkQ  
rBB70GaS2t3fqg//asHYSTBI4IoLibSF5ZJ8NaLo4EqgRts00W1zr8fCoFbxI+yI  
qWriNXR7eoFj8tW07Tj6S0kQr4QZcucLeILfox6CtDz03f3WQTH9m/0si5U4Jf3RA  
gBd0vwxVSSSNEsIUUfy10BHLVw1haTVfh1h1dZhzb18LA1Vqu0100GGxvaG3dU14  
oMSRSK8EeaS04hIM8ScjVyhjsuPRm1j1wd1EXZ5mkHRGRMGmwi3XysJjIeQRFmuG  
a9uPes7I/K85r7X91BLz+G/mkSZsrHxzLxF0mSZtQdhq08GyMeQ1Javs7sb1UVbg  
ag30JEAjgqRsNIQ0MIv12/amrRJ7DUyQu5YkZQsAyCIhSVZw6z1J1kYVKSw1Cu1I  
VX1n/Aahx5wwaQZA1dDTolX8WvL90/KmEGWbKUSov40uoRwS0eXP3Qhw75kd4zT0  
n3pSUc7tXka8GAz5Ar+r9014wc9as2WjWADo3CX8cF0zQ0rN1naMQ++xBIAG9ms8  
y3L4ECoRqvjCpjUAfhflxwSM7uc1CgFBINFKSLV+QGxzWfHjg3+bSQd0hrRn9j4z  
Di9aJmFESQ6iykbUjiUFrcI1NUCFKz2awaxh+Sq8UkYcvux1jxdK/zYYEG8DSdQ2  
uwlssHCjYVpAb16hq1EAAsqnhv86020G4Z6JCg3AUZt9R1MBpK2Pn/NmPSzCwX0E  
EwEKACcCGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AFAL771vAFCQ1nimUACgkQ  
rBB70GaS2t35Jw/9GhpQquJVUCAsc4az7+ad8q2d90UKXOL12x0j3/ofS8umZJYr  
8KXZxPqBqvLj1UyAB5Ur4fWVBDp9iP4Vj74jmrh7YatK8heGmNoUAnI1/90qV50  
ypF0bfvWLxvcUNizmS/LYKdNuYcHwyw4bFyTz9gd3XH6Dxak7YiAXz3JgJIXcIB3  
ELfpsduzpnclEvWz/dKhYX5iJ7Y/xd4Ew8Ian8FyNDNRwcXobTpPAMNiGLG5xSLq  
8RgQfkm07BVVDtu5tPw4V46gnBXGXNjPhSirGMonbKZqtP07TcBQhPQ9c95x73hs  
kAqZKHrwi1B7cfy1I8y5sRFbPa0RXeVWQKEMZdwLsFhCbEex6iXHP+rMK0nSJf0G  
91F2eJk140n8K7wzak0njvN00VfN/9D+xD21CXczbYbaDXX2tY0d4GS72fw9M/zT  
96tIH5UtMGM0ICuUJjnL34EbzbuxwTENJKhDGQH2P+eUzM9jmuCTRLGw2P0Zuof  
6GsSNLf+5pw3BIvEZw5PYT1N6i3m19MQ7p9BWr7f1CF8CU/xzyPN7TVb/677Be7V  
SL1rNfLNi0IdqcbLJ63pTWaUGkCSqRnMfq3cIDlZnZ2LoVv008VVmdtFRkhHN8hJ  
h7CUX1aqB0faJhV3FqA9G8sXmSN3r3pusZb13kfCKsJUZorThWxdaUBuUH3CwX0E  
EwEKACcFALd1gAUCGwMFCQeGH4AFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AACgkQ  
rBB70GaS2t1aKA//dBaXto7zUFRbJvLgcSE3hJ0ChYZj8Uuo7iYK26mVycyBFQJK  
Iv2XYFIJMwK1Rx1Ja1jA6BIKE3aYyx2LVTYU1Hke826dhH+kV2qN9C6t/VmYpV4b  
n/i64po7EzD17ykrm5gB+z47uCVyC79/r80uns1mTf/JUq1/hYJj2hf1MDWr07/X  
Wc1zBLj1T93tg/43Vgz1wFdrU0cNx1+bQGxKT0i4AXSp3UvCL+2YsQ2/JspF7ZzZ  
awSuUVkZJr1lp8751MhZeHmTrCHKV1FHJyudLJErcH0337Jpd8xDRek7K1rx2pAS  
cKIyeAMIk/G10uExYqEEVFQzMr9Z1MNUhNBjAMr5hVGsTgrwy2h1IrBktXav07d  
0leS1sqw9ViZ7F6t90x51+NkwXVsLsGYSzuNyIfomNuoCgA+cfM3TjzVp41qsg1J  
WJc9Bavaan2pKF6Lb9Fq8u3HZk2u+YZbvZkqkXwTdZZQ0kEmoVV4Y1G86bdpmPyj  
8eV7C02NxPii41+qV8qJQu/6DsA0QwMtBMUNODm3BF2+ZmUHuhMGxq9/4vDE8heE  
ffYhHtNftV6JwwzGZmeZkrYA1P9AGLeVp/6iNUe8H5/oPvh4s2rRmqN+L/dQUlix  
i0AT5iAKoRQGkduXrWc4fAY5KxDB9qna4oqX06QP8rEf1I8ELcNgYEj4oJv0wU0E  
V3WABQEQLzM0Cs9Zvd08x0EvbEBj59LrS9d0HVkQ61gmkNakWC+jR35VD6FXpe6  
UYAcBLrEbVYfKw9P0p6MhFKAsb570JoznKGzE1rVYUZQzhD0RKje35rvkajvEcjG  
AWMLTjr87pWHeD0389ER64bz0Rncfa/1+YP56PI+CThb2wUvTTONGJkPQUvVhH+P  
256cQL/Y0Fwu4XLerpwN+YKGMQ47raRcydobPeSfMQr9fVKRy0zFE0rvNpCVDUqi  
77d0gLDLjH11LDy0X5554S8XYLb91eY0iFvnu2pTCKiiExRCSYK29mAQePK1TCCn  
Qx0jbmBbGS8mVIkpQ5vpvXvzpY3JIjMXaDGqWSQSYGXhECyxCR5e0tKYbCwwPIc2  
rI15gW6yXyw9pKmj5XafTP7YHTvRSr7CZ/VLkDkW16AfQ9nPOg1mjwjpdFpmN71h

J1SKMaZkh0QGV5FW3dK+GLwxiWdqx3htbZErvyWvumWQF/xBF7puKJBEXcoM5KfkJ  
uZekBwcnVkfNFF2RdkM1ALq8InGzLXc7R0uEm0BXVirfju7JRtWlb3UhJWCuhRW2  
muyYegSTkag5MduD1IJK37GL8WI1AL65taYgZegUoxHdSaE0ef0hspxuduz8d33z  
UV1WCFhi+r/+BMCQmTRbF8ao7fTC1dGd084DRP6qE/dMT4u0ZEn7ABEBAAHCwXwE  
GAEKACYCGwwWIQT+uSCfLy8/RmSEH1WsEHS4ZpLa3QUCZwAXCwUJEWvKhQAKCRCs  
EHS4ZpLa3XtzD/9dwi1qffv70UTq8w/21jn1owHp09jxP7WHTmPWHE0BW5yFIW1V  
A1gKN6Ym0dw+LvS5W0KJaRnyewUyBxWvZsn6W1b5qzY7nmCOKJpYtuCUPwiqjXWP  
EM8c/v0MojSuwM0XBAViLv0FhgdUrHn1lk962XvWAW++4DXFh2deaV0163IFMRm0  
PNPDAiPwBVqvBANIh2sLRZ5gd1BXwpVrd+x8tzyr69YrN7hutP1CyPEUM9//mcEh  
vFPsbW/i0x/foCE3NXhQm/rSMKecVn5csXBV2J01Mzi+8txYnrSBLkjbSB1AvTQ1  
aG3+nCNCgM2XDLy0j0IrgZ1To4Ay5gmTOR+msY/cfoIuKFYenmtxy6jM8o5uSZHg  
hoClrx9IA98hhGQ73G2r5EDpXuU/uCXn53Sswj65b19IssfqEIoji/FonkpeEgeg  
bGXFduNrhcD0/W0zqpXf2Fa0DQWY+Vc/pt52ftBFgwzCNIUYDKUChPNz0wtLtd  
N2fkXHNiCavCDZ10ud7FHHwmRNdj2q1uKxe4m+pFYmKwAU/H+Htkz9Gjsj+ZKedY  
nnfai2s2gQ0rbfwwV9VdhCWSuLK17ZnGTtiJu0UQI1V8n6QQJpohd3mVgmynu6gQ  
uKw0YS2RuEUFv0v0g2tASA+4EM/SBUpGhud0DLA4b5w04gKmh1B1HqQrIsLBfAQY  
AQoAJgIbDBYhBP65Ij8vLz9GZIQeVawQezhmktrdBQJ1JEokBQkPj/2fAAoJEKwQ  
ezhmktrdwMAP/RpFy1IL4yhgcB0EnQ7e3No80raNk0z/YhSd125N/uQVEU94JGQ  
rrvQ+4L fve21aPweBD018/A0Csm0yHPVQMA0a2vx8ItVdIcNc8iFkP4AJ192210q  
i0Vh0b1UeZnlFK9+Qvq4PQ21hWJr0uzyL/S38REsAT1I25sfJOP+RCaR1MH9dm85  
E56Lee6uZR8SkGuiL6kGpPh6fWTNij3bICjth1iSSCL2HC0W81vcwS1dDu2EfILU  
QCSqfSG7bF8dFk+nKhzhVX0Uks3XGjLdICxZewU5ycryitpfRgARgZs2A43gshdi  
fiKaX6Ksan03uhKDrLhDHNj2y07PurFo8gg1RpV/Pr1B/UqCsC9FU0ixbD+n4ZF  
Sqov2qwe1Lj0f4mZ6yiLsTDU0FPrdk01HTJZ17AF0zXZMM6CvaCUaJCKx9GvdSrR  
+LI4wLQonPrTnXavhkC4intlqSX8ZQNLhEggdE8YwMEJn59R/nVIT3i5WzYph5R9  
P4Vz3Yn7jRqM8wAyEbHkA8s45fMRi9akWSw93H5nWukcmfkt3UEbmka3BQg3HKWP  
6TvhfI28euM8qqjbPilfkpEBjnChYVvk2Rgn0P8zA7Q5kCo293kwJL9c3RDjMPcxI  
45ktKvBTZftsDt1Z718LwW7Q3VQiGiKvo1XLMuV7Z51fmydfUPcrnv17wsF1BBgB  
CgAPAhsMBQJhMqGaBQkLn1UVAaOJEKwQezhmktrdbhwQAITmFb67XIUZswr3TREd  
Q7ZCLG4EDyFTsW8n75r6A90qsR+z68nC2Sm7e8mKQFFPwjHP0hsGhHtCOTZtQk70  
jbwyL4N3uxDyEv0fbckH5Wz0ejZcG7KKQrqAiWJJ7q6CH/zOnVurySjVyzJpy/wL  
WpVAcF/uaW5ZhlFCXqePaEzsUBJ757qsr2ho14BV4seT1RSQ9nneTZ0Hhab3wqXP  
4qdTo8+zktvNo9YbeZ1qj6211+QGIUBTP5MEdXCuC1e4FQ3f6vnXxmB86cUPx7c1  
/y2rIjei0dkKgPeUjNwWSzxS2jYehL5we7gvaSwmEvJ74pV+/3Hs+TxX39XtYFwj  
k9I795idnsS511dAW3yoI3HBQsYa3US7bpH4g3yZMkstc3bHJ6X54PMcd8Skb+N3  
FE8+zGduDmDTKitumiWVvxEFGIwSLAcWPxecI2AMIMGfMheURYsdvD/yvCbCB29  
0KwCSrDvkAG9N2VorNzd7KUEtPTMN1bg2d11F6u5sQeTN5KVaGd7xE10XME2wA2D  
T3+EsAQytriFbcWm3s8Ugbc9BXMmKBfjlvKu6+Fr6Mgvf/txn56M2SyXBCFQ50Ft  
qTFuAFIRv+nayk5tx5Eg1iA7u3dbB1jH3yxGH1B7TeQypA5BqD3x72b7vbXkeci3  
1Kz035LYoT5/yTK5sGvacIvCwsF1BBgBCgAPAhsMBQJgmrz3BQkLBNByAAoJEKwQ  
ezhmktrdLmgP/1FkWkYhxAcnkagRv09mpP12STbu0B3zYKFBALm/Wa7vKDz18dgC  
S3BxDS1pnhZS8QA3Vjmb0AZvaDnsN1UJ0f3Qao5136G/UXPnmFIwN612szP0K6nF  
PEsotzIzR1Jo3S+WkBfiKaQDIDgSxtUxJz0wufz76xibmKRhJ5ChMDCvxmIaoNle  
tKRxFT770upnnyaQs22UsueqrZJ0resgTVnNeF4A1+1U59pFuAlf971SVLr472LP

```

Uj8mPJihF2ukL1Hdz3F7+kY1p0JRmLk9fo4d1ZHBUPiZ1ML/U2yhQfW+Y6tW71vf
izAxJWF7se6QT+UT5Pji6cohMSERVoYt8e2jFjs0PiPcrjU3mJEx4hAEEVIbP9RY
eKC4CL/UGYAtJkUjd85vKZUHYr7NWZQKLAKqpAPQUMKrIKLEHuz/doq2CCamstLI
vcBgg8EjLJn13SBesFt/1DCWZeummqz3omQKR19EHU2cIzIf0Cv/IEysnmbpSpjZ
DX8Fqjtezoq1qiyrLFR7YN1VDPBCHYfqDagw10nlrWFJqT6Vqfs1mdMdTBRWYVEB
0GUxrkyI+APdi0M2634/410b1ptkqyTIr1KIg1J/qsSiKVcBZS0YFW/rskxYcPPT
wpKYaycEYt0dkS6FPcnehJ001B+F32WVq2bs2Ps8we6KhjjaYS4Iv4dwwsF1BBgB
CgAPAhsMBQJe+9W/BQkJZ4k1AAoJEKwQezhmktrdvzMQAI6BBj3c2r4bDpV3TwkX
dQ+UCa/E/zUhFds9XKfGb3a5IzRdPUwT+KrAZyiYrr2NSM0zh1/VtqJL18YCYsx0
0b/TB1hDM+IZiI5gH0cHKhDYKTnNSGP09P/pJAlvHQend9CdZE9J9jwkczfS+bz6
mVxkxpi73fTDox9dues0LsS2/ntRzA0wqhDdaaavRvhAEf9vavCWVrNZmq22WVsU
lnIPxNWGGzWn85JYI6uAi4f4/ABFkry69/c0cvbr0P8qgCmeCuGmX4f0j7qRg77A
+mSueBDx8RK002o1021B7b8IcVizj+1psRQN0oa+i+mFG+o6vtD1ZYhQude4N5sR
RybcLclxjSCoZs5q9JfTpbB2n7pSf/UD3ytwnt9kpD4Vv9dTGAPB83bjL+QK6e3A
XM10jxFE5jSFSr94E40kK80YcIR5jLqsg2f610ENY5drMSA4zuDFDL1Y2ChfjgjZ
uNoFbPHGt/8DfWTV0ochVnikA7ggKjz20+RjvwyrrHhRMAft08MMh9UV28pdL+H53
o0t0V0u5aoTbcNqdYQy9B2Bw4lfmj2fi6Dpl+vnZp6h0m0CWiJVW/dtilppYjuxd
w5Kj+9IxZYaBNYH4l1pMT+BsvMDqGzXxDIL89NnY5BkMvqEKnjXSHGRWYMz0xigf
51YKbfQnEQ1oz5bRQndntRQWwsF1BBgBCgAPBQJXdYAFaHsMBQkHhh+AAAoJEKwQ
ezhmktrdTyEP/0H0VWHwQsaWjMrGj000MFzxGUo8SBmYYTbs29VM8wBGDsPkYCje
ZzU16i9iqDpDqxyqmTigcjhV8CDx/6xsMBLG2yKaKZ4m3+Yn0Qf/sQkyCvqiyMF
9mS7pDYWy+mPhPuw8TDIfiqgVhzjSpIMFWPqxVjn6KKbPN/QASr3Pf0cuP6qpHG+
NAM6Q5dYkCebyvwzLmg1sVnil6iSyJd1jBj3D34XrgWS9buyxBB2CjIM76WxfNVi
J9zAaPI78X9v6PpDGn0kg6oLzrusrvBjoZknKQm0SZ+41fx6xvrTPs8uPEzevzJB
lkke6kw9+KagY8mrVX1ZenRg+sY/4vxJreYwQeq167ggx+wFjKdcfhZA7m70LHOD
ysrGVCLcmuinUBaNLHmLDcGYXZ+kMCoXf0bpuCVByQmNJgEb47EIFlx/+TEeNHKM
0+22xL1atFzXfkEVZck+NghLZyFDhS3g1bma7puU7r752uiJjA6Iv8+kHDXi+/V7
GNpuiEFUYh69QQ2//CS5H51osC/Bkb9evSn/Lp8dMubtWAaXDGJMgw9vqZ55N02N
K0fvF/IKHnGkvH28rv00PCv0WTA/MClv28y0PrSvcmXnduLtkBEX7TISMPW+n+0
Ta63/z4YFFeZ7sFLrEm3Q3vJMN3mE5i3cw+JGXPSu0nTtgqk/oZv//SS
=bboB
-----END PGP PUBLIC KEY BLOCK-----

```

Haltbarkeitsdatum: 08.10.2024

Schlüssel-ID	0 x AC1 07B386692DADD
Typ	RSA
Größe	4096/4096
Erstellt	2016-06-30

Ablaufdatum	2024-10-08
Benutzer-ID	AWS SDKs und Tools < @amazon .com> aws-dr-tools
Schlüssel-Fingerabdruck	FEB9 209F 2F2F 3F46 6484 1E55 0 7B38 692 HINZUFÜGEN AC1

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-----
```

```
xsFNBFd1gAUBEACqbmFbxdJgz1lD7wr1skQA1LLuSAC4p8ny9u/D2zLR8Ynk3Yz
mzJuQ+Kfjne2t+xTDex6MPJ1MYp0viSwsX2psgvdmeyUpW9ap01rThNYkc+W5fRc
buFehfbi9LSATZGJi8RG0sCCr5FsYVz0gEk85M2+PeM24cXhQIOZtQUjswX/pdk/
KduGtZASqNAYLKR0mRODzUuaokLPo24pfm9bnr1RnRtw5ktPAA5bM9ZZaGKriej
kT2lPffBjP8F5AZvmGLtNm2Cmg4FKBvI04SQjy2jjrQ3wBzi5Lc9HTxDuHK/rtV
u6PewUe2WP1nx1XenHMZU1UK4YoSB9E9StQ2VxQiySLHSdxR7Ma4WgYdVLn9b0ie
nj3QxLuQ1ZUKF79ES6JaM4t0z1gGcQeU1+Uk1gjFLuKwmzWRdEIFFxMyvH6qgKnd
U+DioH5mcUwhwffAAsuIJyAdMIEUYh7IfzJJXQf+fF+Xf0C16by0JFWrIGQkAzMu
CEvaCfwtHC2Lpzo33/WRFEmauzzd0QJ4uz4xFFvaS0SZHMLHWI9YV/+Pea3X99Ms
0Nlek/LolAJh67MynHeVB0HKIrq+fLuoRwepQivctzN6Y1N0kx5naTPGgaKWK7G2q
TbcY5SMnkIWfLFSouj0Fvmjczq8iZRwYxWA+i+LQvsR9WEXEiQffIWRoQARAQAB
zsFNBFd1gAUBEAC8zNArPwb3dPMTThL2xAY+fS60vXdB1Sk0tYJpDwPfgvo0d+VQ+
hV6Xu1GAHAS6xG1WHysPT9KejIRSGLG+e9CaM5yhsxNa1WFGUM4Q9ESo3t+a75Go
7xHIxgFjC046/06Vh3g9N/PREeuG8zkZ3H2v5fmD+ejyPgk4W9sFL00zjRiZD0FK
VYR/j9uenEC/2NBcLuFy3q6cDfmCoDE0062kXMnaGz3knzEK/X1SkcjsxRDq7zaQ
lQ1Kou+3dICwy4x5SJQ8j1+eeeEvF2C2/dXmDohb57tqUwioohMUQkmCtvZgEHjy
pUwgp0MTo25gWxkvJlSJkU0b6b1786WnySIzF2gxq1kkEmB14RAssQkeXjrSmGws
MDyHNqyJeYFus18sPaSpo+V2n0z+2B070Uq+wmf1S5A5FpegH0PZZoNZo8I6Qxa
Zje9YSZUijGmZIdEBleRVt3Svhi8MYlnasd4bW2RK1sr7p1kBf8QRe6biiQRF3KD
OSn5CbmXpAchJ1ZHRRdkXZDNQC6vCJxsy1300TrhJtAV1Yq347uyUbVi291ISVg
roUVtprsmHoEk5Go0THbg9SCSt+xi/FiJQC+ubWmIGXoFKMR3UmhDnnzobKcbnbs
/Hd981FdVghYYvq//gTAKJk0WxfGq030wtXRndPOA0T+qhP3TE+LtGRJ+wARAQAB
wsF1BBgBCgAPBQJXdYAFaHsMBQkHhh+AAAoJEKwQezhmktrdTyEP/0H0VWHwQsaW
jMrGj00MMFzxGUo8SBmYYTBS29VM8wBGDsPkYCjeZzU16i9iqDpDqxyqmTigcjh
V8CDx/6xsMBLG2yKaKZ4m3+Yn0Qf/sQkyCvqiyMF9mS7pDYWy+mPhPuw8TDIfiqg
VhzjSpIMFWPqxVjn6KKbPN/QASr3Pf0cuP6qpHG+NAM6Q5dYkCebyvwzLmg1sVni
16iSyJd1jBj3D34XrgWS9buyxBB2CjIM76WxfNViJ9zAaPI78X9v6PpDGN0kg6oL
zrusrvBjoZknKQm0SZ+41fx6xvrTPS8uPEzevzJB1kke6kw9+KagY8mrVX1ZenRg
```

```
+sY/4vxJreYWQeq167ggx+wFjKDcfhZA7m70LH0DysrGVCLcmuinUBaNIHmLDcGY
XZ+kMCoXf0bpuCVByQmNJgEb47EIF1x/+TEeNHKM0+22xL1atFzXfkEVZck+NghL
ZyFDhS3g1bma7puU7r752uiJjA6Iv8+kHDXi+/V7GNpuiEFUYh69QQ2//CS5H51o
sC/Bkb9evSn/Lp8dMubtWAaXDGJMgw9vqZ55N02NK0fvF/IKHnGkvH28rv00PCv0
WTA/MClv28y0PrSvcmXnduLtkBEX7TISMPW+n+0Ta63/z4YFfEZ7sFLrEm3Q3vJ
MN3mE5i3cw+JGXPSu0nTtgqk/oZv//SS
=Z9u3
-----END PGP PUBLIC KEY BLOCK-----
```

# Dokumentverlauf

Diese Seite listet wichtige Änderungen am AWS SDK für Java Developer Guide im Laufe seiner Geschichte auf.

Dieser Leitfaden wurde veröffentlicht am: 1. Oktober 2025.

1. Oktober 2025

Fügen Sie einen neuen [PGP-Schlüssel hinzu, der](#) am 27.09.2026 abläuft.

5. Oktober 2024

Aktualisieren Sie die [aktuellen OpenPGP-Schlüsselinformationen](#).

4. September 2024

Fügen Sie Informationen zu AWS kontobasierten Endpunkten für DynamoDB hinzu. Siehe [the section called "Verwenden Sie AWS kontobasierte Endpunkte"](#).

21. Mai 2024, 2024

Entfernen Sie die Anweisungen zum Einstellen der `networkaddress.cache.ttl` Sicherheitseigenschaft mithilfe einer Java-Befehlszeilen-Systemeigenschaft. Siehe [Wie legt man die JVM-TTL fest](#).

12. Januar 2024

Fügen Sie ein Banner hinzu, das das Ende der Unterstützung für AWS SDK für Java v1.x ankündigt.

6. Dezember 2023

- Geben Sie den [aktuellen OpenPGP-Schlüssel](#) an.

14. März 2023

- Aktualisierung des Leitfadens zur Ausrichtung an bewährten IAM-Methoden. Weitere Informationen finden Sie unter [Bewährte IAM-Methoden](#).

28. Juli 2022

- Es wurde eine Warnung hinzugefügt, dass EC2 -Classic am 15. August 2022 in den Ruhestand geht.

22. März 2018

- DynamoDB Beispielsweise wurde die Verwaltung von Tomcat-Sitzungen entfernt, da dieses Tool nicht mehr unterstützt wird.

## 2. Nov. 2017

- [Kryptografiebeispiele für Amazon S3 Verschlüsselungsclients](#) hinzugefügt, einschließlich neuer Themen: [Verwenden Sie clientseitige Verschlüsselung und Amazon S3 clientseitige Verschlüsselung mit verwalteten AWS KMS-Schlüsseln und clientseitige Verschlüsselung mit Amazon S3 Client-Hauptschlüsseln.](#)

## 14. April 2017

- [Der AWS SDK für Java Abschnitt „Amazon S3 Beispiele zur Verwendung des Buckets“](#) wurde aktualisiert, darunter neue Themen: [Verwaltung von Amazon S3 Zugriffsberechtigungen für Buckets und Objekte und Konfiguration eines Buckets als Website. Amazon S3](#)

## 4. April 2017

- Ein neues Thema, [Enabling Metrics for the](#), AWS SDK für Java beschreibt, wie Anwendungs- und SDK-Leistungsmetriken für die AWS SDK für Java generiert werden.

## 3. April 2017

- Dem AWS SDK für Java Abschnitt „ CloudWatch Beispiele verwenden“ wurden neue [CloudWatch Beispiele](#) hinzugefügt: [Metriken abrufen aus CloudWatch](#), [Veröffentlichen benutzerdefinierter Metrikdaten](#), [Arbeiten mit CloudWatch Alarmen](#), [Verwenden von Alarmaktionen in CloudWatch](#) und [Senden von Ereignissen an CloudWatch](#)

## 27. März 2017

- Weitere Amazon EC2 Beispiele wurden [dem AWS SDK für Java Abschnitt „Amazon EC2 Beispiele verwenden“](#) hinzugefügt: [Amazon EC2 Instanzen verwalten](#), [Elastic IP-Adressen verwenden in Amazon EC2](#), [Regionen und Availability Zones verwenden](#), [Mit Amazon EC2 Schlüsselpaaren arbeiten](#) und [Mit Sicherheitsgruppen arbeiten in Amazon EC2.](#)

## 21. März 2017

- [Neue IAM-Beispiele](#) wurden zu den IAM-Beispielen hinzugefügt. [Verwenden Sie den AWS SDK für Java Abschnitt: Verwaltung von IAM-Zugriffsschlüsseln, Verwaltung von IAM-Benutzern, Verwendung von IAM-Kontoaliesen, Arbeiten mit IAM-Richtlinien und Arbeiten mit IAM-Serverzertifikaten](#)

## 13. März 2017

- [Dem Amazon SQS Abschnitt](#) wurden drei neue Themen hinzugefügt: [Aktivieren von Long Polling für Amazon SQS Nachrichtenwarteschlangen](#), [Einstellen des Sichtbarkeits-Timeouts in und Verwenden von Warteschlangen für unzustellbare Nachrichten in. Amazon SQS Amazon SQS](#)

## 26. Januar 2017

- [Es wurden ein neues Amazon S3 Thema, Using TransferManager for Amazon S3 Operations, und neue Best Practices für die AWS Entwicklung hinzugefügt, wobei das AWS SDK für Java Thema im Abschnitt Verwenden des Themas enthalten ist. AWS SDK für Java](#)

## 16. Januar 2017

- Es wurden ein neues Amazon S3 Thema, [Verwaltung des Zugriffs auf Amazon S3 Buckets mithilfe von Bucket-Richtlinien](#), und zwei neue Amazon SQS Themen, [Arbeiten mit Amazon SQS Nachrichtenwarteschlangen](#) und [Senden, Empfangen und Löschen von Amazon SQS Nachrichten](#), hinzugefügt.

## 16. Dezember 2016

- Es wurden neue Beispielthemen hinzugefügt für DynamoDB: [Arbeiten mit Tabellen in DynamoDB](#) und [Arbeiten mit Elementen](#) in. DynamoDB

## 26. Sept. 2016

- Die Themen im Abschnitt „Erweitert“ wurden in „[Verwenden von](#)“ verschoben [AWS SDK für Java, da sie für die](#) Verwendung des SDK von zentraler Bedeutung sind.

## 25. August 2016

- Ein neues Thema, [Creating Service Clients](#), wurde dem Abschnitt [Verwenden von](#) hinzugefügt, [in dem](#) gezeigt wird AWS SDK für Java, wie Client Builder verwendet werden können, um die Erstellung von AWS-Service Clients zu vereinfachen.

Der Abschnitt mit den [AWS SDK für Java Codebeispielen](#) wurde mit [neuen Beispielen für S3](#) aktualisiert, die von einem [Repository](#) unterstützt werden GitHub, das den vollständigen Beispielcode enthält.

## 02. Mai 2016

- Dem AWS SDK für Java Abschnitt [Verwenden wurde ein neues Thema, Asynchrone Programmierung, hinzugefügt, in dem](#) beschrieben wird, wie mit asynchronen Client-Methoden gearbeitet wird, die Future Objekte zurückgeben oder die eine annehmen. `AsyncHandler`

## 26. Apr. 2016

- Das Thema SSL-Zertifikatanforderungen wurde entfernt, da es nicht mehr relevant ist. Unterstützung für SHA-1-signierte Zertifikate wurde im Jahr 2015 als veraltet markiert und die Website mit den Test-Skripts wurde entfernt.

## 14. März 2016

- Dem Amazon SWF Abschnitt [Lambda-Aufgaben](#) wurde ein neues Thema hinzugefügt, in dem beschrieben wird, wie ein Amazon SWF Workflow implementiert wird, der Lambda Funktionen als Aufgaben aufruft, als Alternative zur Verwendung herkömmlicher Amazon SWF Aktivitäten.

## 04. März 2016

- Der AWS SDK für Java Abschnitt „[Amazon SWF Beispiele zur Verwendung des](#) Themas“ wurde mit neuen Inhalten aktualisiert:
  - [Amazon SWF Grundlagen](#) — Enthält grundlegende Informationen darüber, wie Sie SWF in Ihre Projekte integrieren können.
  - [Eine einfache Amazon SWF Anwendung erstellen](#) — Ein neues Tutorial mit step-by-step Anleitungen für Java-Entwickler, die noch keine Erfahrung damit haben Amazon SWF.
  - [Aktivitäts- und Workflow-Worker ordnungsgemäß herunterfahren](#) — Beschreibt, wie Sie mithilfe der Parallelitätsklassen von Java Amazon SWF Worker-Klassen ordnungsgemäß herunterfahren können.

## 23. Februar 2016

- Die Quelle für das AWS SDK für Java Entwicklerhandbuch wurde verschoben nach. [aws-java-developer-guide](#)

## 28. Dezember 2015

- [the section called “Legen Sie die JVM-TTL für die Suche nach DNS-Namen fest”](#) wurde von Advanced in [Using the verschoben und aus Gründen der AWS SDK für Java](#) Übersichtlichkeit neu geschrieben.

[Verwenden des SDKs mit Apache Maven](#) wurde mit Informationen über die Einbindung der SDK-Bill of Materials (BOM) in Ihr Projekt aktualisiert.

## 04. August 2015

- SSL-Zertifikatsanforderungen sind ein neues Thema im Abschnitt „[Erste Schritte](#) AWS“, in dem die Umstellung auf SHA256 -signierte Zertifikate für SSL-Verbindungen beschrieben wird. Außerdem wird beschrieben, wie Java-Umgebungen mit Version 1.6 und früheren Versionen repariert werden können, sodass diese Zertifikate verwendet werden können, die für den AWS Zugriff nach dem 30. September 2015 erforderlich sind.

### Note

Java 1.7+ ist bereits in der Lage, mit SHA256 -signierten Zertifikaten zu arbeiten.

## 14. Mai 2014

- Das Material zur [Einführung](#) und [zu den ersten](#) Schritten wurde stark überarbeitet, um die neue Struktur des Leitfadens zu unterstützen. Es enthält nun auch Anleitungen zur [Einrichtung von AWS Zugangsdaten und zur Region für die Entwicklung](#).

Die Besprechung der [Codebeispiele](#) wurde in ihr eigenes Thema im Abschnitt [Zusätzliche Dokumentation und Ressourcen](#) verschoben.

Informationen zum [Anzeigen des SDK-Revisionsverlaufs](#) wurden in die Einführung verschoben.

## 9. Mai 2014

- Die allgemeine Struktur der AWS SDK für Java Dokumentation wurde vereinfacht, und die Themen [Erste Schritte](#) und [Zusätzliche Dokumentation und Ressourcen](#) wurden aktualisiert.

Neue Themen wurden hinzugefügt:

- [Mit AWS Anmeldeinformationen arbeiten](#) — Erläutert die verschiedenen Möglichkeiten, wie Sie Anmeldeinformationen für die angeben können AWS SDK für Java.
- [Zugriff auf AWS Ressourcen mithilfe von IAM-Rollen gewähren auf Amazon EC2](#) — enthält Informationen zur sicheren Angabe von Anmeldeinformationen für Anwendungen, die auf EC2 Instances ausgeführt werden.

## 9. Sept. 2013

- In diesem Thema, dem Dokumentverlauf, werden die Änderungen am AWS SDK für Java Entwicklerhandbuch nachverfolgt. Es handelt sich um ein begleitendes Dokument zu release notes history (Verlauf der Versionshinweise).